Semantic Controllable Image Generation in Few-shot Settings

Jianjin Xu

Submitted in partial fulfillment of the
requirements for the degree of
Master of Science in Computer Science
under the Executive Committee
of the Fu Found School of Engineering and Applied Science

COLUMBIA UNIVERSITY

2021

Abstract

Semantic Controllable Image Generation in Few-shot Settings

Jianjin Xu

Generative Adversarial Networks (GANs) are able to generate high-quality images, but it remains difficult to explicitly specify the semantics of synthesized images. In this work, we aim to better understand the semantic representation of GANs, and thereby enable semantic control in GAN's generation process. Interestingly, we find that a well-trained GAN encodes image semantics in its internal feature maps in a surprisingly simple way: a linear transformation of feature maps suffices to extract the generated image semantics.

To verify this simplicity, we conduct extensive experiments on various GANs and datasets; and thanks to this simplicity, we are able to learn a semantic segmentation model for a trained GAN from a small number (e.g., 8) of labeled images. Last but not least, leveraging our findings, we propose two few-shot image editing approaches, namely Semantic-Conditional Sampling and Semantic Image Editing. Given an unsupervised GAN and as few as eight semantic annotations, the user is able to generate diverse images subject to a user-provided semantic layout, and control the synthesized image semantics.

Table of Contents

Acknowledgements

My first thanks go to my advisor Changxi Zheng. Changxi has always influenced my research style with his rigor, devotion, and persistence in science. The most important thing I learned from him is to be responsible for my research, and only do research that makes me feel pride.

I appreciate Shuran Song and Carl Vondrick for serving on my thesis committee. Shuran has always encouraged me to do research, and her kindness helped me a lot throughout my research. Attending her course was also a great pleasure. Carl has long inspired me through his contributions to Computer Vision and novel ideas on self-supervision. It is my honor to have him on my committee.

Many great minds led me into and through the journey of AI research. Feng Chen led me into the world of computer vision. He was not only my research mentor but also a life instructor when I was a freshman. Xiaolin Hu supervised me for a long time and taught me valuable research principles. I am honored to work with him on many projects and look forward to working with him in the future. In the 2018 summer, I became an intern in MSRA and worked with Xun Guo and Yan Lv. Their incisive insights and systematical way of research helped me a lot to form a more mature research style.

Many colleagues are crucial in my research experience. Among them, members from Tsinghua-Microsoft Future Internet and Technology club are special to me. Weiran He, the club chairman in 2016, guided me into research and provided many opportunities for my personal growth. Shaoxiong Wang, the club chairman in 2017, taught me not only research skills but also experience in running a club. Encouraged by Shaoxiong, I started co-managing the FIT club starting from 2018 and became the chairman in 2019, which turned out to be my valuable experience. Kun Wu was the co-chair at that time, and he also helped and inspired me greatly. During my chairman period, I built good friendships with Fan Jin, Yuhao Huo, and Kaichao You. I also had a pleasant cooperation experience with Yuekai Jia and Mingyang Kou on club projects. I cannot be more appreciative of their help and inspiration.

I deeply appreciate other senior mentors and colleagues, especially Ceyuan Yang, Zheyang Xiong, Yunyun Jiang, Lovish Chum, Yujun Shen, Yinghao Xu, Kun Yu, Jiarui Fang, Yanwen Wei, Shuyao Kong, Xiahui Ben, Ke Sun, Feng Lin, Yulong Wang, Yueqiao Li, Chang Xiao, Peilin Zhong, Jian Wu, Fangzhou Liao.

I enjoyed spending time with my friends. Thanks to Yixuan Wang, Haoyu Qin, Ke Yang, Yang Zhao, Qianrui Zhang, Huajun Wu, Fangdan Ye, Chao Yao, Hua Deng, Zongyi Li, Feng Ji, Tingbo Lv, Changran Hu, Ruoran Shi, Muchen Yuan, Runqi Zhu, Yuchen Mo, Weishu Mo, Yue Qin, Bohan Qu, Yilun Wang, Junrui Zhu, Xinchun Yang, Yifan Zhang, Pai Liu, Kaiwen Xu, Zhiwen Fan, Wenting Cai, Feng Ji, Lan An, and Tao Luo, for the games, parties, and conversations.

Finally, I deeply thank my parents for their unconditional love and support.

Dedication

To Xiaojing

Chapter 1: Background and Introduction

1.1 Introduction

Recently, Generative Adversarial Networks (GANs) [1] have achieved tremendous success in various image synthesis applications, such as high-resolution faces [2], diverse bedroom images [3], outdoor scenes [4, 5], etc. However, their ability to synthesize images conditioned on semantic constraints is still limited. Typically, unsupervised GANs [4, 6, 7] cannot sample images having a certain attributes specified by a user.

Increasing research interests have been directed towards how to **control** GAN's image generation. Researchers propose new architectures that have better controllability [8, 9, 10] and methods to exploit controllability [11, 12, 13] from pretrained GAN models. The controllability of categories and attributes has achieved great success, while controlling for an image's spatial structure is still challenging. The task of synthesizing image conditioning on spatial information is usually addressed in the image-to-image context [14, 15, 16, 17]. These methods allow explicit control of semantic structures but generally need expensive labeled data for training. For example, in order to synthesize an image from a semantic mask, pix2pix [15] needs a paired image-annotation dataset, and the cycleGAN [14] also needs a dataset of semantic masks. As the image-semantics relation is difficult to model, image-to-image methods require expensive semantic labeling datasets to learn.

However, unsupervised GANs are also speculated to have modeled image-semantics well, as suggested by many works [12, 11, 18, 19]. Are we able to decode the semantics modeled by an unsupervised GAN and thereby enable its controllability for generated image semantics?

This thesis first study how to **decode semantic masks** from the feature maps of a generator. Although GANs are believed to have modeled image semantics, the semantic encoding rule remains unclear. The most related work by far is GAN Dissection [18], which identifies causal units able to manipulate certain semantic concepts in the generated image. However, GAN Dissection cannot decide the exact category of image pixels. Instead of finding interpretable units, we examine all the feature maps collectively to build a precise semantic segmentation of a generated image. Surprisingly, we find that a simple linear transformation, named Linear Semantic Extractor (LSE), suffices to extract the image semantics.

To verify LSE's competence for decoding semantics, we conduct experiments on various GAN architectures and datasets. We choose three of the most widely used GAN architectures: Progressively Grown GAN [2] (PGGAN), StyleGAN [3], and StyleGAN2 [7], trained on CelebAHQ [20], FFHQ [7], or LSUN's bedroom and church datasets. The performance of LSE is further compared to nonlinear semantic extraction methods to verify whether LSE suffices to extract the generated image semantics. We propose two Nonlinear Semantic Extractors (NSE-1 and NSE-2) for comparison. The image semantics extracted by the LSE, NSE-1, and NSE-2 are compared to semantic masks predicted by a pretrained segmentation network (UNet for facial images, and DeepLabV3 for bedroom and church images). The performance of semantic extractors is measured by mean Intersection-over-Union (mIoU, defined in Section A.2), which is a common metric in semantic segmentation context. Results show that although NSEs have higher mIoU, the margins are minimal: for most GAN models and datasets, LSEs' relative performance drop relative to NSEs' is within 3.5%. We also provide evidence from geometrical perspectives that **(i)** feature vectors of the same category are clustered, and **(ii)** features of different categories are well-separated measured by cosine similarity. Therefore, it is well-backed that GANs indeed use a linear notion to embed semantics.

The training of LSE is supervised, where the supervision is provided by off-the-shelf segmentation networks. Interestingly, the simplicity of LSE suggests that the **few-shot** learning of LSE might be possible, which we refer to as the few-shot LSE. We find that a LSE trained with as few as 16 annotations are able to get close to their fully supervised counterpart. For example, in StyleGAN2-FFHQ, the 16-shot LSE achieves 88.1% performance relative to its fully trained version. Not only does the success of few-shot LSEs backs GAN's linear semantics embedding, but also inspires new approaches for controlling image generation under few-shot settings.

We consider two important image editing applications: **(i)** Semantic Image Editing (SIE) and **(ii)** Semantic-Conditional Sampling (SCS). The former aims to update images based on the user's edit on the semantics of a GAN's output. For instance, generate images where the hair region is reshaped according to user's edits. The latter is meant to generate images subject to a user specification of desired semantic layout. For example, produce images of a bedroom where the furnitures are laid out according to user's specification. We demonstrate few-shot SIE and SCS algorithm both rely only on only a small number of annotated images.

Both SIE and SCS are built with a core idea: to match a target by optimizing generator's latent vector. Essentially, both SIE and SCS are formalized as latent vector optimization problems. The target of SIE is to change an existing latent vector to match a modified semantic mask. SCS's target is to find suitable latent vectors that match a given semantic mask. To evaluate SIE and SCS, we consider baselines for both tasks, which use a pretrained segmentation network instead of a few-shot LSE. In comparison to the baselines, our approach with 8-shot LSE is able to generate comparable (and sometimes even better) results.

In summary, our contributions are twofold: **(i)** Through extensive experiments, we show that GANs represent the image's pixel-level semantics in a linear fashion. **(ii)** We propose an LSE with few-shot learning, which further enables two image synthesis applications with semantic control, namely SCS and SIE under few-shot settings.

1.2 Related Works

1.2.1 Generative Adversarial Networks

GAN [1] is composed of a generator network G and a discriminator network D. The generator maps a randomly sampled noise z to an image, and the discriminator tries to tell whether the image is real or fake. By playing against each other, the generator learns to generate images that are indistinguishable from the discriminators. Another variant of GAN [15] use image-like data for the input of generator, which can translate an image I_A in one domain to image I_B in another domain. For example., the image translation could be from “spring” to “autumn”, from “horse” to “zebra”, from “semantic masks” to “real images”, and etc. We refer to these two types of GANs as Noise-to-image GANs (N-GANs) and Image-to-image GANs (I-GANs), for simplicity.

N-GANs have achieved tremendous success in synthesizing various images, such as faces, birds, cars, and bedroom scenes [4, 21, 2, 6, 22, 3]. The resolution of generation has been improved to 1024×1024 for faces [2, 7] and 512×512 for ImageNet [23] images [24, 25]. Various losses and training methods are also proposed to improve the generation quality of GANs [26, 27, 28]. Among various N-GAN models, progressively Grown GAN (PGGAN) [2], StyleGAN [3], and its improved version StyleGAN2 [7] are three of the most widely used architectures. PGGAN shares a similar architecture as the Deep Convolution GAN (DCGAN) [4] but is trained progressively. StyleGAN adopts the adaptive instance normalization [29] from neural stylization literature and improves the generation quality on many tasks. StyleGAN2 is improved upon StyleGAN and is currently the state-of-the-art GAN model on various datasets. At the time of this work, these three GAN models are sufficiently representative for most N-GANs. Therefore, we choose to conduce experiments on them.

I-GANs can also synthesize photo-realistic images. The pix2pix model [15] can transform images from one domain to another domain given a paired dataset. CycleGAN [14] relaxes the pairing constraint by introducing the cycle consistency loss. [30] improves the image resolution and quality for I-GANs. It can also transform semantic masks to 1024×1024 HD cityscapes [31] images. [16, 17] continue to improve the precision of the mapping from semantic masks to images. However, if we use I-GANs to instantiate semantic-conditional image synthesis, a large and densely annotated dataset is required. The labeling cost would be prohibitive if users need semantic controllable image synthesis on a new dataset. Therefore, we choose not to consider I-GANs in this thesis.

1.2.2 Interpreting GANs

This thesis tries to decode the semantics of the generator, which is related to interpreting or dissecting GANs. Methods on this topic can be grouped into two categories. First, interpreting the latent space of GANs. [11, 12] found that there are linear boundaries on latent space, separating positive and negative image samples of attributes. [32, 13, 33] propose unsupervised methods to find the linear trajectories of attributes in the latent space. Second, interpreting the feature maps of GANs. GAN Dissection [18] identifies convolution units that have causality with semantics in the generated images. By manipulating the causal units, the semantics of the corresponding region can be edited. [34] found semantic clusters by k-means and matrix factorization in the features of GAN. This thesis studies the relation between generator's feature maps and generated image's semantics. Our differences are two-fold. First, the semantics studied is of high resolution and accuracy. Second, our aim is few-shot image editing application, which is not touched by existing methods.

1.2.3 Controlling GANs

Researchers have also explored methods to enable the controllability of N-GANs, and these methods often have low labeling costs. There are two major approaches for this purpose. First, training new GANs with architecture designed to enable controllability. Second, exploiting the controllability which has already been modeled by pretrained N-GANs.

Methods focusing on designing **new** architectures is mostly concerned with global attributes controllability. Conditional GAN (cGAN) and its variant [9, 35] are proposed to enable category controllability for N-GANs. In cGANs, a vector describing category information is concatenated with the latent vector together to form the input of GAN models. Supervised by class labels, cGANs learn to sample images of desired categories. InfoGAN [8] further learns the category-level controllability without manual labeling. StackGAN [10] extends cGAN by using the embedding of natural language to control the synthesis. An interesting trial refers to [5], which explores using scene graphs to control scene generation, yet the image quality is limited.

Methods focusing on exploiting the controllability of **pretrained** N-GANs can be further classified by the types of controllability they enabled. First, the controllability of **global attributes**. Many interpretation-based editing methods fall into this category [11, 12, 32, 13, 33]. Abdal et al. [36] construct Conditional Continuous Normalizing Flow to manipulate images' attributes. Second, the controllability of **3D characteristics**. [37, 38] make use of 3D models to control the pose and lighting of faces. Zhang et al. [39] control the camera pose of synthetic car images. Thirdly, the controllability of **localized editing**, which refers to edit a localized region of an image realistically without changing other regions. Zhu et al. [40, 41] use a latent code optimization pipeline to make the generated image resemble color strokes drawn by users. Though they can adjust the images locally to some extend, the precision is limited. [34, 42] propose feature map collaging, which is to substitute features in target feature maps for features selected from source images.

The semantic controllability studied in this thesis differs two-fold. First, previous methods on Semantic-Conditional Sampling require expensive labeling, while we focus on using only a few annotations for this task. Second, Semantic Image Editing is concerned with changing the morphology of objects rather than changing a localized region freely. In practice, the user may use local editing methods to change the morphology of objects, but the interaction might not be as straightforward as SIE.

1.3 Thesis Structure

Chapter 1 introduces the background, related work, and the structure of the thesis. In Chapter 2, we study how to decode GAN's encoding of semantics. We propose the Linear Semantic Extractor and describe its training and few-shot learning methods. Two Nonlinear Semantic Extractors (NSEs) are proposed for comparison. In Chapter 3, we conduct experiments to show that the LSE suffices to decode GAN's semantic encoding. We first describe the experiment and evaluation setup. Secondly, we present the qualitative and quantitative results of LSEs and NSEs. Thirdly, few-shot LSEs are trained and evaluated. In Chapter 4, we present the few-shot SCS and few-shot SIE. The details of experiment setup and additional results are placed in the appendix.

Chapter 2: Decode GAN's encoding of semantics

This chapter aims to decode GAN's internal representation of image semantics in its image synthesis process. Our finding is surprisingly simple: a **linear transformation** on the GAN's feature maps suffices to reveal its synthetic image semantics. We first construct such a linear transformation (also referred to as the Linear Semantic Extractor, LSE). Then, two Nonlinear Semantic Extractors (NSEs) are proposed for comparison.

2.1 Linear Semantic Extractor

A well-trained GAN model maps a randomly chosen latent vector to a realistic image. Structurally, a GAN model concatenates a series of network layers. Provided a latent vector, each layer i outputs a feature map $\mathbf{x}_i$, which is in turn fed into the next layer. We denote the width, height, and depth of $\mathbf{x}_i$ using w_i, h_i and c_i, respectively (i.e., $\mathbf{x}_i \in \mathbb{R}^{c_i \times w_i \times h_i}$).

It is unsurprising at all that one can deduce from the feature maps the generated image semantics. After all, feature maps represent the GAN's internal data flow that results in the final image. As images can be segmented using pretrained networks, the feature map can also be segmented with appropriate networks. More interesting is the question of how **easily** we can learn from feature maps about the generated image semantics. A straightforward relation between feature maps and image semantics could inspire new theories and applications.

Consider a GAN model consisting of N layers and producing images with m semantic classes (such as hair, face, and cloth). We seek the simplest possible relation between its feature maps and output image semantics — a linear transformation matrix $\mathbf{T}_i$ applied to each feature map $\mathbf{x}_i$ to predict a semantic map of the layer i. By accumulating all the maps, we wish to predict a semantic segmentation $\mathbf{S}$ of the GAN's output image (see Figure 2.1). Formally, $\mathbf{S}$ is just a linear

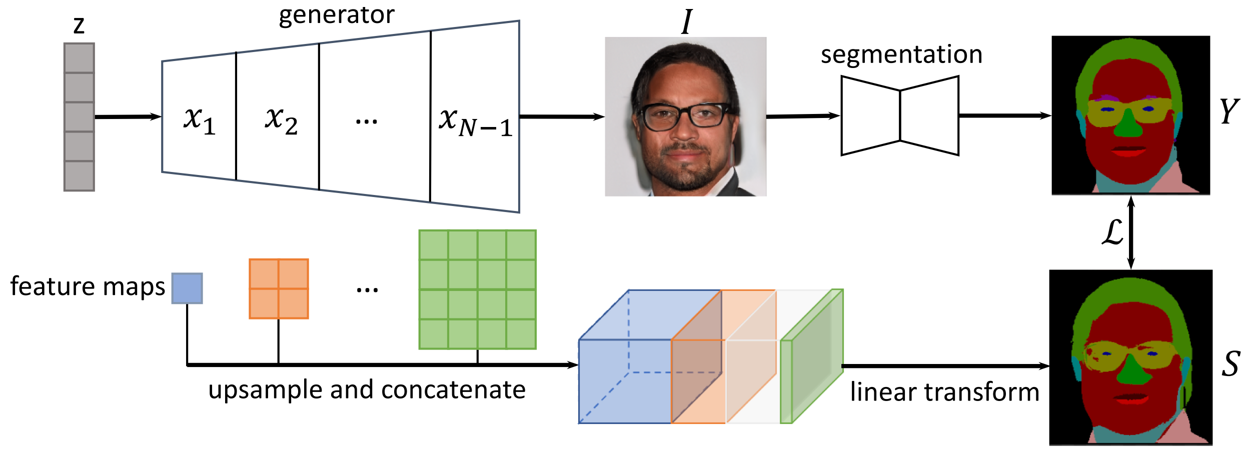


Figure 2.1: When synthesizing an image I from a latent vector z, the generator builds a series of feature maps $\{\mathbf{x}_i\}_{i=1}^{N-1}$. We decode those feature maps to semantic segmentation S using a linear transformation, also referred to as the Linear Semantic Extractor (LSE). The LSE is supervised by a pretrained segmentation model.

transformation of all feature maps, defined as

$$\mathbf{S} = \sum_{i=1}^{N-1} \mathsf{u}_i^{\uparrow}(\mathbf{T}_i \cdot \mathbf{x}_i), \tag{2.1}$$

where $\mathbf{T}_i \in \mathbb{R}^{m \times c_i}$ converts $\mathbf{x}_i \in \mathbb{R}^{c_i \times w_i \times h_i}$ into a semantic map $\mathbf{T}_i \cdot \mathbf{x}_i \in \mathbb{R}^{m \times w_i \times h_i}$ through a tensor contraction along the depth axis. The result from each layer is then upsampled ($\mathsf{u}_i^{\uparrow}$) to the output image resolution. The summation extends over all internal layers, excluding the very last layer (layer N), which outputs the final image. The result $\mathbf{S} \in \mathbb{R}^{m \times w \times h}$ has the same spatial resolution $w \times h$ as the output image. Each pixel $\mathbf{S}_{ij}$ is a $m \times 1$ vector, indicating the pixel's unnormalized log probabilities representing each of the m semantic classes. This method is abbreviated as Linear Semantic Extractor (LSE).

In the next two sections, we will describe how to learn the parameters of the LSE in fully supervised settings and few-shot settings, respectively.

2.1.1 Training with full supervision

The training of the LSE requires pixel-level annotation of semantics. As we are concerned with synthetic images from a GAN model, it is impractical to manually annotate a training dataset formed by images sampled from GANs. Therefore, we use the semantic masks predicted by pre-trained off-the-shelf segmenters for supervision. In practice, we use UNet [43] for segmenting faces, and DeepLabV3 [44] with ResNeSt backbone [45] for the segmentation of the bedroom dataset and the church dataset.

Concretely, provided a well-trained GAN model, we randomly sample its latent space to produce a set $\mathbb{S}$ of synthetic images. When synthesizing every image in $\mathbb{S}$, we also record the model's feature maps $\{\mathbf{x}_i\}_{i=1}^{N-1}$. These feature maps are linearly transformed using (2.1) to predict a semantic mask of the image, which is then compared with the result from the pretrained semantic segmentation network to form the standard cross-entropy loss function:

$$\mathcal{L} = \frac{1}{w \cdot h} \sum_{\substack{1 \le i \le w \\ 1 \le j \le h}} \left[-\mathbf{S}_{ij}[Y_{ij}] + \log \left(\sum_{k=1}^{m} \exp\left(\mathbf{S}_{ij}[k]\right) \right) \right] \tag{2.2}$$

where Y_{ij} is the semantic class of pixel (i, j) indicated by the supervisor network, and $\mathbf{S}_{ij}[k]$ is the corresponding unnormalized log probability of the k-th semantic class predicted by the LSE (2.1).

Lastly, the linear matrices $\mathbf{T}_i$ are optimized by minimizing the expected loss (estimated by taking the average loss over image batches in $\mathbb{S}$). The details of the training are placed in Section 3.1.

2.1.2 Training in few-shot settings

Generally, few-shot learning makes use of knowledge learned from previous tasks to ease the learning of the new task. Here, we rely on the semantic knowledge learned by GANs. Assume that GANs do encode semantics linearly, it is likely that we can train LSE with a few annotations given its simplicity. The intuitions are two-fold: **(i)**, the linearity makes the whole optimization process convex, thus the model parameters are easy to learn. **(ii)**, the linear model is not likely to overfit according to our assumption because the linear form matches the semantic encoding rule. The assumption will be tested in Chapter 3.

However, the few-shot settings bring up an inherient deficit: the training data points might not cover all the modes of GANs. For example, the eyeglasses category is rare in generated images, and the few-shot LSE is likely to get no supervision at all for this category. But this is not a real bottleneck in practice because users can repeatedly sample images until the desired rare category appears. During this process, the users do not need to annotate more images and thus the labeling cost is still low.

In a real-world scenario, the supervision of few-shot learning is assumed to be manual annotations. To be specific, consider the following scenario: a user trains a GAN on an unpopular dataset and wants to control the image semantics of the GAN model. The user manually annotates a few images and trains a few-shot LSE. In expectation, the few-shot LSE can segment the images reasonably well and can support SCS and SIE proposed in this work. For simplicity, we choose to simulate human annotations with pretrained segmentation network.

Let the number of annotations be N, and the total training iterations be M. The steps of training a few-shot LSE are:

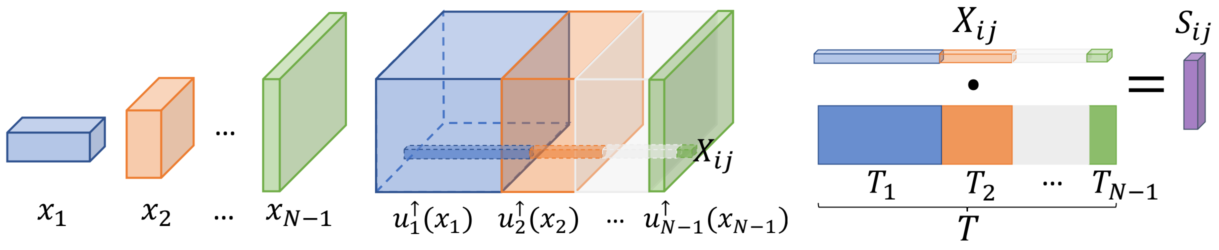
1. Sample N latent vectors and segment their images using the pretrained segmenter.
2. Optimize $\mathbf{T}_i$ using the loss function (2.2) on all the latent vectors.
3. Repeat step 2 for M iterations. For StyleGAN and StyleGAN2, new layer noises are sampled in every iteration.

In order to evaluate few-shot LSEs more rigorously, the above process is repeated five times for each model to avoid training data's variance.

2.1.3 Geometric Interpretation

The linear relation (2.1) allows us to draw an intuitive geometric picture of how image semantics are encoded in the generator's feature maps.

First, notice that $\mathbf{T}_i$ applied on $\mathbf{x}_i$ can be viewed as a 1×1 convolutional filter with stride 1. The filter operation is commutative with the upsample operation $\mathsf{u}_i^{\uparrow}(\cdot)$ (A proof is placed in



(a) feature maps $\mathbf{X}$ (b) upsampled feature maps $\mathbf{X}$ (c) dot product $\mathbf{S}_{ij} = \mathbf{T} \cdot \mathbf{X}_{ij}$

Figure 2.2: Illustration of the linear transformation (2.1).

Section A.1). Thus, we can rewrite the semantic prediction $\mathbf{S}$ as

$$\mathbf{S} = \sum_{i=1}^{N-1} \mathbf{T}_i \cdot \mathsf{u}_i^{\uparrow}(\mathbf{x}_i) = \mathbf{T} \cdot \mathbf{X}, \tag{2.3}$$

where $\mathbf{T} = \begin{bmatrix} \mathbf{T}_1 & \dots & \mathbf{T}_{N-1} \end{bmatrix}$ is an $m \times n$ matrix with $n = \sum_{i=1}^{N-1} c_i$ being the total layer depth. $\mathbf{X} \in \mathbb{R}^{n \times w \times h}$ is a tensor concatenating all upsampled $\mathbf{x}_i$ (i.e., $\mathsf{u}_i^{\uparrow}(\mathbf{x}_i)$ with resolution $c_i \times w \times h$) along the depth axis (see Figure 2.2b).

Now, consider a pixel (i, j) in the output image. To predict its semantic class, equation (2.3) shows that we can take the corresponding $n \times 1$ vector $\mathbf{X}_{ij}$ that stacks the pixel's features resulted from all GAN layers, and dot product it with each row of $\mathbf{T}$ (see Figure 2.2c): $\mathbf{S}_{ij} = \mathbf{T} \cdot \mathbf{X}_{ij}$. In other words, each row $\mathbf{T}^{(k)}$ of $\mathbf{T}$ defines a direction representing the semantic class k in the n-dimensional feature space.

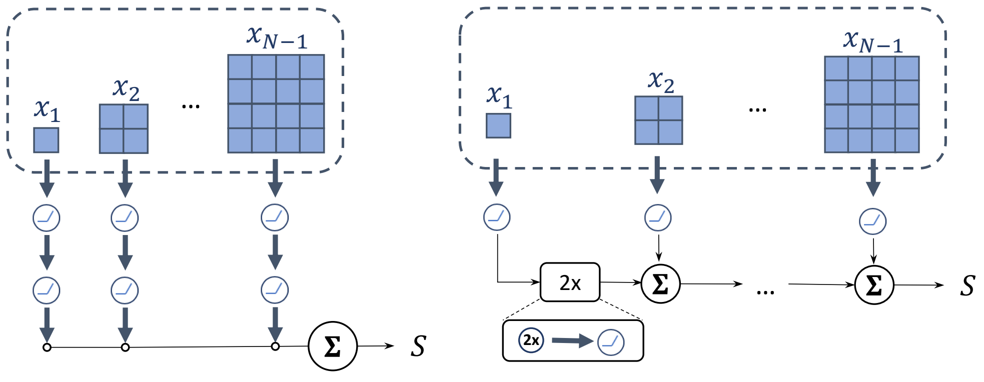
If the linear transformation can classify features with high accuracy, it indicates that the feature vectors of different semantic categories are linearly separable. Consider the set of all vectors that are classified into category k, i.e., the class score for k should be larger than scores for other classes. The set $\mathcal{R}_k$ is defined as:

$$\mathcal{R}_k = \{\boldsymbol{x} | \mathbf{T}^{(k)}\boldsymbol{x} > \mathbf{T}^{(j)}\boldsymbol{x}, \forall j \neq k\} \tag{2.4}$$

where $\mathbf{T}^{(k)}$ is the k-th row of the tensor $\mathbf{T}$. This definition shows that the subspace of each semantic class forms a hyper-cone originating from the origin.

A geometric picture is as follows: Consider a unit n-sphere at the origin. The intersection of a semantic class k's hyper-cone and the sphere surface encloses a convex area A_k. Then, take the feature values at a pixel and normalize it into a unit vector. If that vector falls into A_k, then this pixel is classified as class k. As a result, the hyper-cones for M classes completely divide the n-sphere into M convex areas, each representing a semantic category. From this geometric perspective, we can even infer a pixel's semantic class without training the linear model (2.1). We locate a representative center c_k for each convex area A_k on the n-sphere surface. For example, the semantic centers can be estimated by a clustering algorithm (such as k-means clustering). A pixel is classified as class i if its feature vector is closest to c_i (among all semantic centers) on the n-sphere. In Section 3.2.2, we show that the class centers can segment images reasonably well, supporting our hyper-cone interpretation.

2.2 Nonlinear Semantic Extraction



(a) NSE-1 (b) NSE-2

Figure 2.3: The architecture of NSEs. The thick blue arrow refers to 3×3 convolution with stride 1. The rectangle "2x" block refers to the upsample-convolution block, where the circle "2x" refers to nearest upsampling with factor 2.

If the linear transformation extracts generated image semantics plausibly, a further question is to what extent the semantics can be better extracted by nonlinear transformations. Generally, the nonlinear transformations will more accurately extract the semantics because they have a larger representation capacity than the linear transformations. Nevertheless, the performance loss of the linear method provides further support on whether or not the feature maps in GANs encode image semantics linearly. If they indeed encode semantics in a linear way, nonlinear models would perform **not** significantly better than our linear model.

To explore this question, we propose two nonlinear extraction models. The architectures of NSEs are shown in Figure 2.3. The first nonlinear semantic extractor (NSE-1) transforms the feature map of each layer through three convolutional layers with ReLU activations in-between. It then upsamples each feature maps using the same interpolation $\mathsf{u}_i^{\uparrow}(\cdot)$ as in (2.1). Formally, the semantics extracted by NSE-1 is calculated as

$$\mathbf{S}_{\text{NSE-1}} = \sum_{i=1}^{N-1} \mathsf{u}_i^{\uparrow}(\mathbf{H}(\mathbf{x}_i)) \tag{2.5}$$

where $\mathbf{H}$ is the nonlinear convolution layers. NSE-1 is a direction generalization from LSE. For each layer, the only difference is that the NSE-1 uses nonlinear convolution layers while the LSE uses a single linear convolution.

The second model (NSE-2) transforms feature maps into hidden layers and refines them as the resolution increases. It adopts the widely used “upsample-convolution” block and resembles the architecture of DCGAN [4].

One may also examine a nonlinear transformation in extreme — for example, one that concatenates a generative model to a full-fledged semantic segmentation model (such as UNet [43]). However, such a model provides no insight into how feature maps encode image semantics. Therefore, an over-complex model is undesirable for decoding GAN’s encoding of semantics, and we choose not to consider those extreme models for our purpose.

2.3 Summary

In this chapter, we propose linear and nonlinear semantic extractors to decode GAN’s encoding of semantics. The LSE uses a linear transformation as specified in (2.1) to transform the feature maps of the generator into semantic masks. The fully supervised training and few-shot training of LSE are proposed. We also introduce the geometric interpretation of LSE. Two NSEs are also proposed for comparison.

In the next chapter, we will experiment to prove the sufficiency of LSE for decoding.

Chapter 3: Experiment

In this chapter, we provide empirical evidence that LSE suffices to extract the image semantics of GANs. First, we introduce the experiment setup and evaluation methods. Second, we present qualitative and quantitative results of LSEs and NSEs. Third, we show evidence from geometric perspectives to support the linear semantic relation.s

3.1 Experiment Setup

Pretrained models. We make our experiment as generalizable as possible by conducting experiments on various GANs and datasets. We choose Progressively Grown GAN (PGGAN) [2], StyleGAN [3], and StyleGAN2 [7] trained on the face, bedroom, and church datasets. The face dataset is CelebAHQ [20] for StyleGAN and FFHQ [3] for StyleGAN2, which is the same as the official releases. The bedroom dataset and church dataset are subsets in LSUN [46] dataset. All of the pretrained GAN models are obtained here [1].

The training of semantic extractors relies on supervisions provided by pretrained segmentation networks. For segmentation on facial images, we train a UNet with 15 classes on CelebAMask-HQ [47], which has manually labeled segmentations. For the bedroom and church images, we use the publicly released DeepLabV3 [44] with ResNeSt [45] backbone trained on ADE20K [48] dataset. The DeepLabV3 model has 150 classes, in which most categories do not exist in the synthetic images. To remove categories that are not present in GAN's generated images, we perform a category selection procedure. First, we train and evaluate LSE, NSE-1, and NSE-2 on the full 150 classes. Then, we remove categories that are predicted with mIoU < 10% by all the semantic extractors. Finally, we train and evaluate the models on the selected categories again in the same settings. We tried to use frequency-based metrics to select categories, but found that this method

[1] https://github.com/genforce/genforce

did not work well. The selected categories and the category-wise mIoU are shown in the appendix (Table A.1 and Table A.2).

Training. For fully supervised training, we synthesize 51,200 images using the GAN and record their feature maps. These images are then semantically segmented by an off-the-shelf segmenter. The semantic masks and feature maps are then used to train the transformation matrix $\mathbf{T}_i$ for every GAN layer. To be specific, the total matrix $\mathbf{T}$ (defined in (2.3)) for StyleGAN2-FFHQ are of size 15×5568. For StyleGAN2-Bedroom, $\mathbf{T}$ is shaped as 16×5376.

$\mathbf{T}_i$ are optimized with Adam [49] with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and initial learning rate 10^{-3}. The training takes 50 epochs in total, where each epoch consists of 1,024 samples. The learning rate is reduced by 10 after 20 epochs. For the first two epochs, the batch size is 1. For the next 16 epochs (3 to 19), the batch size is set to 4. For epoch 20 to 50, the batch size is 64. The total optimization iterations are $1024 \times 2 + \frac{1024}{4} \times 16 + \frac{1024}{64} \times 32 = 6,656$. LSE, NSE-1, and NSE-2 are trained in the same settings.

For the few-shot training of LSEs, we also sample the latent space and segment the images. The difference is that only a few annotations are made available. We experimented with 1, 4, 8, 16 samples, resulting in one-shot, 4-shot, 8-shot and 16-shot LSEs, respectively. For the one-shot LSE, the training takes 2000 iterations with batch size 1. For 4, 8, and 16 samples, the training uses batch sizes 4, 8, and 16 and iteration numbers 2000, 1000, and 500, respectively. For PGGAN, each batch is exactly the same. For StyleGAN and StyleGAN2, the layer noises are re-sampled for each batch. The optimizer setting is the same as in full supervision.

Evaluation. Conventionally, semantic segmentation methods are evaluated on real image-segmentation datasets. However, our semantic extractors cannot take real images as input. One may invert real images in GAN's representation, but the inversion is another challenging problem, thus we do not consider this approach. As a result, the evaluation cannot be conducted on the common annotated dataset. Ideally, we should annotate synthetic images manually, but the cost would then be prohibitive. Therefore, we choose to use the prediction from the off-the-shelf segmenter as the ground-truth for evaluation.

We sample and segment another 10,000 images for evaluation. Every time GAN generates an image, we apply the semantic extractor to the generator's feature maps to predict a semantic mask. The segmentation is compared with the pretrained segmenter's prediction to compute the IoU. The definition of IoU is also placed in the appendix (Section A.2).

As some datasets (e.g., LSUN's bedroom dataset) may be more difficult to segment than some others (e.g., the CelebAHQ dataset), we compute relative performance differences between semantic extractors. Concretely, for each GAN model, there are three semantic extractors to be evaluated, which are LSE, NSE-1, and NSE-2. Denoting their mIoUs with the pretrained segmenter as y_i, and the highest mIoU among the three as y^*, the relative performance difference of each semantic extractor is defined as $\frac{y_i - y^*}{y^*}$.

3.2 Results

3.2.1 Evaluation of LSE

Figure 3.1 compares qualitatively semantic segmentation of LSE to other methods. The quantitative results in terms of mIoU scores are reported in Table 3.1, from which it is evident that our simple LSE is comparable to more complex, nonlinear semantic classifiers. The relative performance gap between LSE and NSEs (NSE-1 and NSE-2) is within 3.5%. Results on StyleGAN-Bedroom and StyleGAN-Church have a slightly larger gap ($< 8\%$). We present additional qualitative results and IoU for each category in Section A.4.

Our experiments show that LSE is capable of extracting image semantics from the feature maps of the GANs. Further, the close performance of LSE to NSEs suggests that a well-trained GAN encodes the image semantics in its feature maps in a linear way.

The training and evaluation of few-shot LSEs are identical to fully supervised LSEs except for the number of training samples. We experiment with 1, 4, 8, 16 annotations obtained from the segmenter. For each few-shot LSE model, the training is repeated five times as the training data variance might be significant. We also compare the performance of few-shot models relative to the

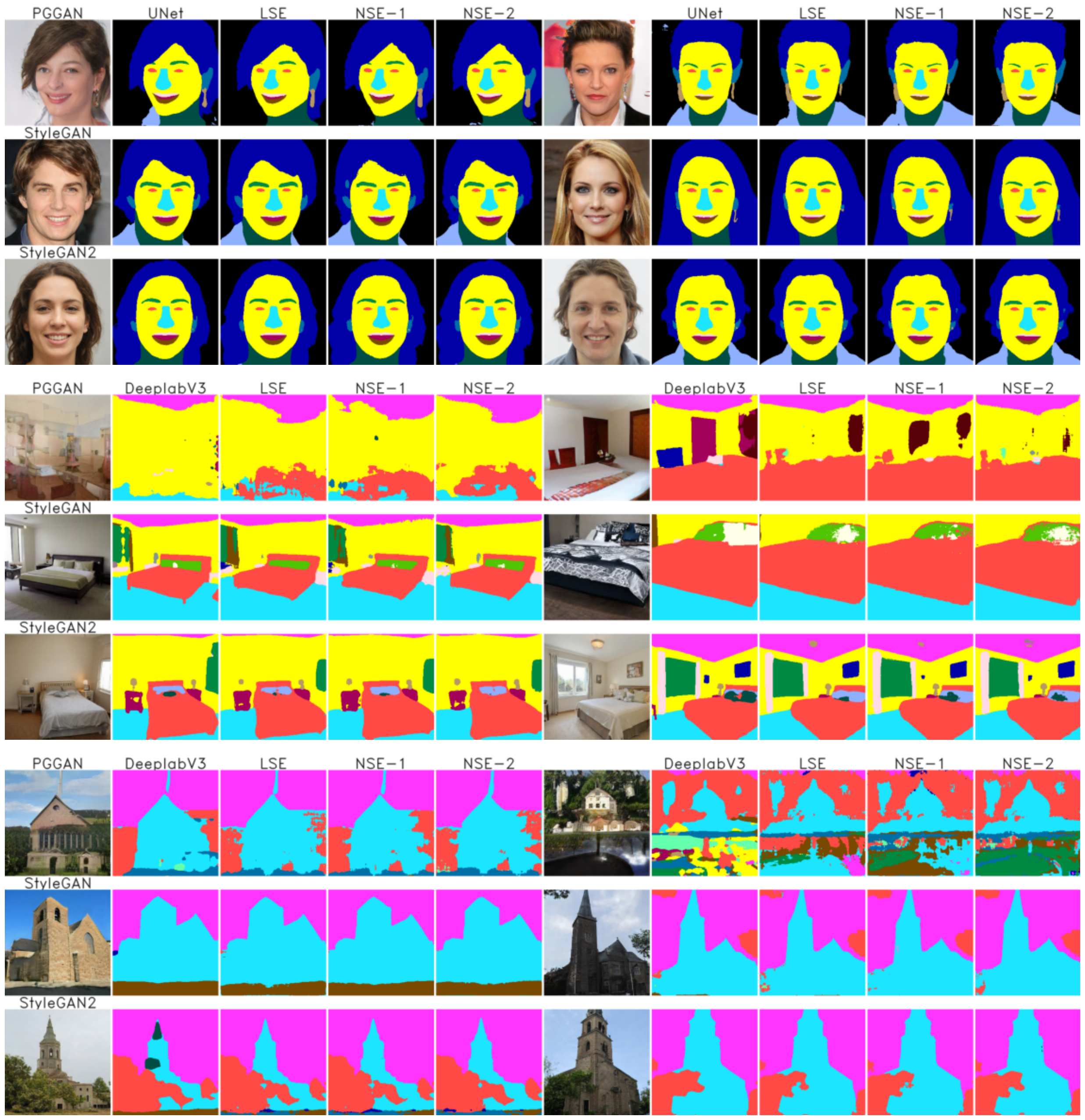


Figure 3.1: Qualitative comparison of LSE, NSE-1 and NSE-2. From top to bottom, every 3 rows are from GAN models trained on the same dataset (face, bedroom, church images, respectively). Images are sampled randomly rather than cherry-picked.

	PGGAN			StyleGAN			StyleGAN2		
Dataset	CelebAHQ	Bedroom	Church	CelebAHQ	Bedroom	Church	FFHQ	Bedroom	Church
LSE	65.5 (-1.6)	33.2 (-3.2)	51.3 (-3.2)	69.1 (-1.9)	39.9 (-7.8)	35.4 (-6.3)	79.7 (-1.7)	53.9 (-3.4)	37.7 (-2.6)
NSE-1	**66.5**	**34.3**	**53.0**	**70.5**	**43.3**	**37.8**	**81.0**	**55.8**	**38.7**
NSE-2	65.9 (-0.9)	30.7 (-10.5)	49.5 (-6.6)	70.1 (-0.5)	38.9 (-10.2)	34.0 (-10.1)	80.2 (-1.1)	52.1 (-6.8)	35.3 (-8.8)

Table 3.1: The mIoU (%) of LSE, NSE-1, and NSE-2 trained with off-the-shelf semantic segmentation models (UNet for CelebAHQ and FFHQ, DeepLabV3 for bedroom and church dataset). "Bedroom" and "Church" images are subsets of the LSUN [46] dataset. The numbers in brakets are the performance difference relative to the best model highlighted in bold.

N	FFHQ	Bedroom	Church
1	55.6 (69.8) ± 5.2	21.5 (39.8) ± 3.7	19.7 (52.2) ± 3.4
4	64.8 (81.4) ± 1.0	36.5 (67.8) ± 2.7	24.2 (64.3) ± 1.4
8	68.4 (85.8) ± 2.6	38.6 (71.6) ± 2.4	26.3 (69.7) ± 0.8
16	70.2 (88.1) ± 3.0	42.2 (78.3) ± 1.1	27.7 (73.5) ± 0.8
full	79.7%	53.9%	37.7%

Table 3.2: The evaluation of few-shot LSEs for StyleGAN2. Each model is trained 5 times. Both the mean and maximum deviation of the 5 repeats are shown. The numbers in parentheses indicate the ratio of the mean performance over the fully trained model's performance listed in the last row.

fully supervised models.

Table 3.2 reports the quantitative evaluation results. First, the extreme case, one-shot LSE, already shows plausible performance, achieving 69.8%, 39.8%, and 52.5% mIoU scores relative to the fully trained model. The 16-shot LSE further improves the mIoU scores to 88.1%, 78.3%, and 73.5% relative to the fully trained model.

3.2.2 Geometry Interpretation

As mentioned in Section 2.1.3, the geometric interpretation of (2.1) indicates that features of a semantic class fall into a convex surface area on an n-sphere. To verify this intuition, we test two stronger hypothesis in this section: **(i)** the features of individual pixels can be clustered around class centers. If the clusters are well formed, we should be able to find a convex hull to identify individual classes. **(ii)** the features of different classes are well-separated under cosine similarity. If the cosine distances between different classes are larger than within the same class, then finding a hyper-cone separating different classes should be easy.

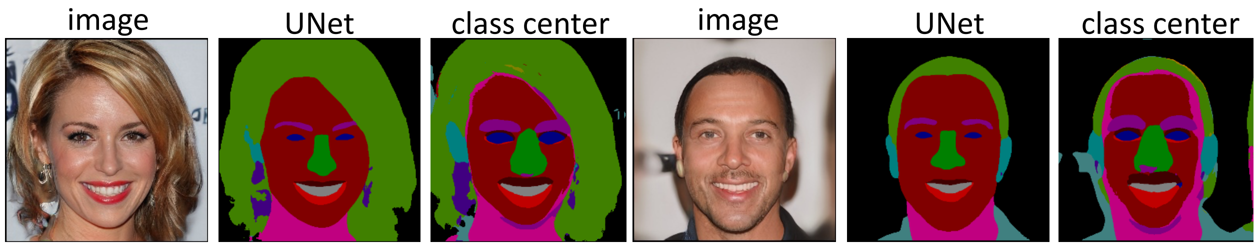


Figure 3.2: Forging LSE's parameter **T** using the statistical centers of features. Experiment is done on StyleGAN-CelebAHQ.

Class centers of features. To estimate the class centers, we randomly generate 3000 images using StyleGAN-CelebAHQ, and obtain their semantic masks using UNet. All per-pixel feature vectors from the same semantic class are collected and normalized onto the unit n-sphere. The vectors are then averaged and renormalized on the n-sphere. The resulting vector is then treated as a class center to determine each pixel's semantic class. Some segmentation results are shown in Figure 3.2, suggesting that this approach indeed segments images reasonably. The segmentation error (e.g., in Figure 3.2) may be attributed to the inaccurate boundaries between classes, as they are not explicitly trained to separate different semantic classes. Nevertheless, this experiment confirms our geometric intuition about the feature maps' linear embedding of semantics.

Cosine similarities between categories. Our purpose is to test whether the distances of features within a category are closer than those between different cateogories or not.

The core part is to sample features for each category fairly, and compute the cosine distances between features. For this purpose, we propose a fair sampling algorithm (Algorithm 1) which repeatedly samples images and record features fairly until enough features are collected. In every image, if the feature number of a category is larger than a threshold T_1, then T_1 feature vectors from that category are chosen randomly without replacement (denoted by $choice(a, N)$). The chosen vectors would be accumulated to a category feature pool until the feature number reach T_2. The algorithm would terminate when all the category feature pools have collected T_2 features. The fair sampling algorithm gauruantees that each category feature pool consists of T_1 randomly chosen vectors from $\frac{T_2}{T_1}$ randomly sampled images. As the sampling procedure is identical for each

Algorithm 1: Fair feature sampling algorithm.

```
Input: $G$; $P$; $T_1$; $T_2$
Output: $\{f_k\}$
for $k = 1, 2, \dots, M$ do
    $f_k = \emptyset$
while $\exists k, |f_k| < T_2$ do
    $z \sim \mathcal{N}(0, I)$
    $I, F = G(z)$ // $F$ denotes features
    $S = P(I)$
    for $k = 1, 2, \dots, M$ do
        if $|f_k| < T_2$ and $|\{p|S_p = k\}| \geq T_1$ then
            $R = choice(\{p|S_p = k\}, T_1)$
            $f_k = f_k \cup \{F_p|p \in R\}$
```

category, the sampled features are fair for each category. In practice, we choose $T_1 = 200$ and $T_2 = 4000$.

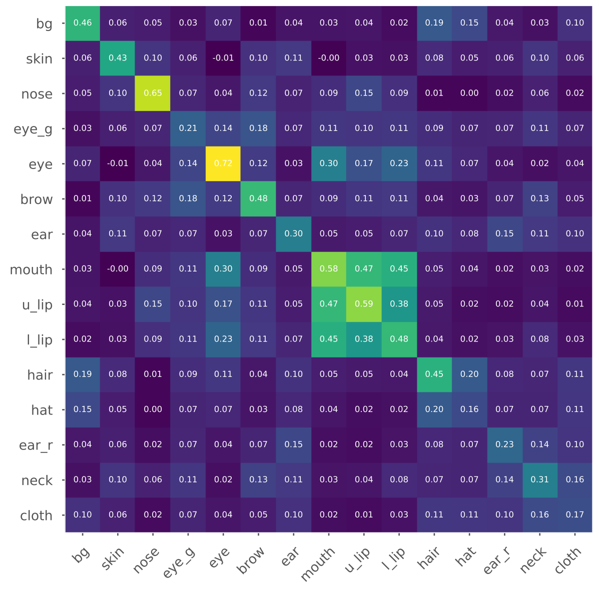
Cosine similarities between categories. We further verify our reasoning by computing the statistics of cosine similarities for feature vectors within the same semantic class and across different classes.

First of all, to ensure the fairness of comparison for each category, we propose a fair sampling algorithm in Algorithm 1, where the choice (a, N) means to choose N samples without replacement from a randomly. The fair sampling algorithm repeatedly samples images and predicts semantic masks using a pretrained segmenter. For each sampled image and each category present in the image, if the total number of vectors belonging to a category is larger than T_1, then T_1 vectors from them are chosen randomly without replacement. The selected vectors are accumulated to a feature pool. The feature pool stops accumulating for one category once it has collected T_2 features in total. When all the categories have collected T_2 features, the algorithm stops. In practice, we choose $T_1 = 200$ and $T_2 = 4000$. This means that each category's features are formed by $\frac{4000}{200} = 20$ randomly selected pixels from each image in 200 randomly sampled images. Therefore, for each category, their features are sampled in an identical way.

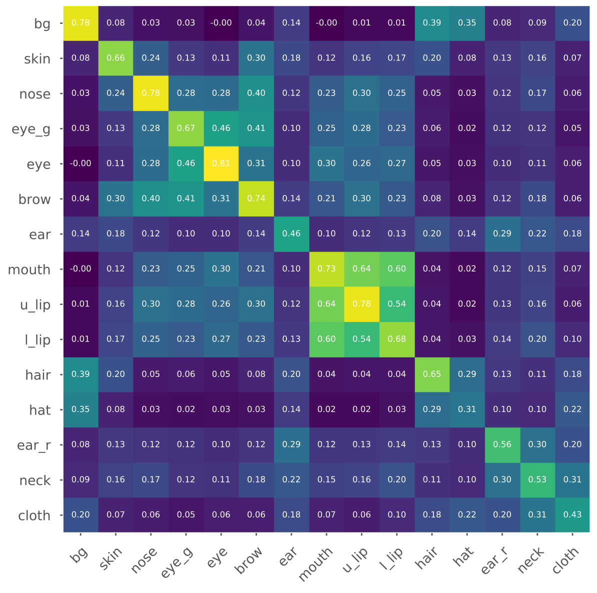
Second, we calculate the cosine similarity between categories using the fairly sampled features.

Specifically, we first calculate the pairwise cosine similarity between feature vectors of two pools, resulting in a $T_2 \times T_2$ confusion matrix. The two pools can belong to different categories (inter-class) or the same category (intra-class). The cosine similarity $cos(A, B)$ between A and B is defined as the mean of the entire matrix.

We show results of StyleGAN-CelebAHQ and StyleGAN2-FFHQ in Figure 3.3. Most diagonal elements of the confusion matrix have higher cosine similarity than other elements in a row. It is indicated that the features in a category are indeed more similar to one another than features between different categories.



(a) StyleGAN-CelebAHQ



(b) StyleGAN2-FFHQ

Figure 3.3: The cosine similarity between categories for GANs trained on face dataset. The features for each category are collected using Algorithm 1.

3.3 Summary and Remarks

In this section, we conduct experiments to support LSE's sufficiency for extracting semantics from the generator. First of all, we evaluate LSE, NSE-1, and NSE-2 and observe that the relative performance drop of LSEs compared to NSEs are mostly within 3.5%. Second, we propose few-shot learning of LSEs, which achieves performance comparable to fully supervised LSEs with as few as 16 annotations. Lastly, we present geometrical evidence that features in the same category are clustered and features in different categories are well-separated. In conclusion, it is backed that GANs indeed use a linear notion to embed semantics.

Our approach differs from the prior work GAN Dissection [18] that identifies units (slices along depth axis) correlating with a specific semantic class. Most units are found in middle-level feature maps, resulting in a lower resolution than the network output. Also, they did not study the per-pixel category belonging and thus form semantic masks. In contrast, the semantics extracted by LSE are of high resolution (the same as the output image) and have sharp boundaries. Notice that there is no conflict between our results and GAN Dissection: having some units correlated with the semantic concept indicates that we can linearly combine them to get more accurate segmentation. Another prior work by Collins et al. [34] clusters features of a particular GAN layer for image editing purpose. They use unsupervised KMeans on a single layer to get rough semantic masks, while we use supervised and few-shot learning method on all GAN layers to get precise semantic segmentations.

The simplicity of the linear semantic embedding not only enables a low-cost way of extracting semantics from GANs, but also inspires novel image editing applications.

Chapter 4: Applications

We have shown that it is possible to train LSE in few-shot settings. In this chapter, we leverage the simplicity of LSE to control the image semantics in GAN's generation process.

4.1 Few-shot Semantic Image Editing

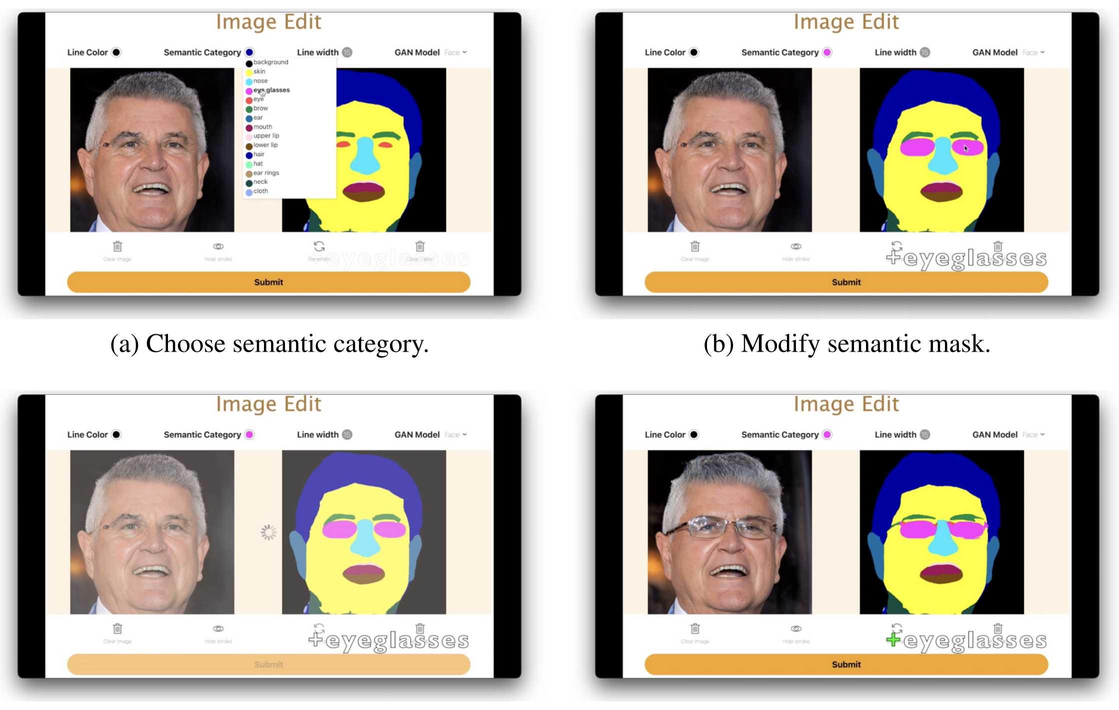
Algorithm. In many cases, the user may want to control a GAN's image generation process. For example, they might want to adjust the hair color of a generated facial image from blond to red; and the user may draw a red stroke on the hair to easily specify their intent. Existing approaches, such as color space editing [40, 41, 50, 51], aim to find a latent vector that generates an image better matching the user specification. The latent vector is often found by minimizing a differentiable distance between the generated image and the user's strokes in color space. The general pipeline of the optimization formulation can be summarized by Algorithm 2, whose inputs are the generator G, the edit loss L, the optional regularization loss L_{reg}, and total iteration number N.

For color space editing, the editing loss will be the color editing loss L_c, defined as $L_c = \frac{1}{||M||_2^2}||M \odot (G(z_i) - C)||_2^2$, where C is the color stroke, M is the mask of the modified region. However, without explicit notion of semantics, the minimization process may not respect image semantics, leading to undesired changes of shapes and textures. For example, in the 2nd row and 2nd and 3rd columns of Figure 4.2, the user wishes to remove the hair in generated images, but the color space editing methods tend to just lighten the hair color rather than removing it.

Leveraging LSE, we propose an approached called Semantic Image Editing (SIE) to enable semantic-aware image generation. We define a semantic edit loss $L_s = \mathcal{L}(P(G(z)), Y)$, where $\mathcal{L}(\cdot)$ is the cross-entropy loss, Y is the target semantic mask, and P is a pretrained segmentation model such as our LSE. Starting from an image's latent vector z, we find an output image's latent

Algorithm 2: Image editing algorithm.

```
Input: $G$; $L$; $L_reg$; $N$
Output: latent code $z$
for $i = 1, \ldots, N$ do
    $z_i \leftarrow z_{i-1} + \text{optimizer}(L(z_{i-1}) + L_{\text{reg}}(z_{i-1}))$
$z \leftarrow z_N$
```



(a) Choose semantic category.

(b) Modify semantic mask.

(c) Submit to server.

(d) Result returned from server.

Figure 4.1: The web application for Semantic Image Editing. The user stroke data is collected using this portal.

vector z' by minimizing the loss.

In practice, we also add a regularization loss composed by items including the color preservation loss $L_p = \frac{||(1-M)\odot(G(z_i)-G(z_0))||_2^2}{||1-M||_2^2}$, the neighbor regularization loss $L_n = ||z_i - z_0||_2^2$, and the prior regularization loss $L_z = ||z_i||_2^2$. z_i denotes the latent vector for the i-th iteration and z_0 denotes the initial latent vector. For color space editing, its total loss is $L = L_s + 10^{-3}L_n + 10^{-3}L_z$. For SIE, the total loss is $L = L_c + L_p + 10^{-3}L_n + 10^{-3}L_z$. We use Adam as the optimizer with default parameters. The optimization repeats for 50 iterations with a learning rate fixed to be 0.01.

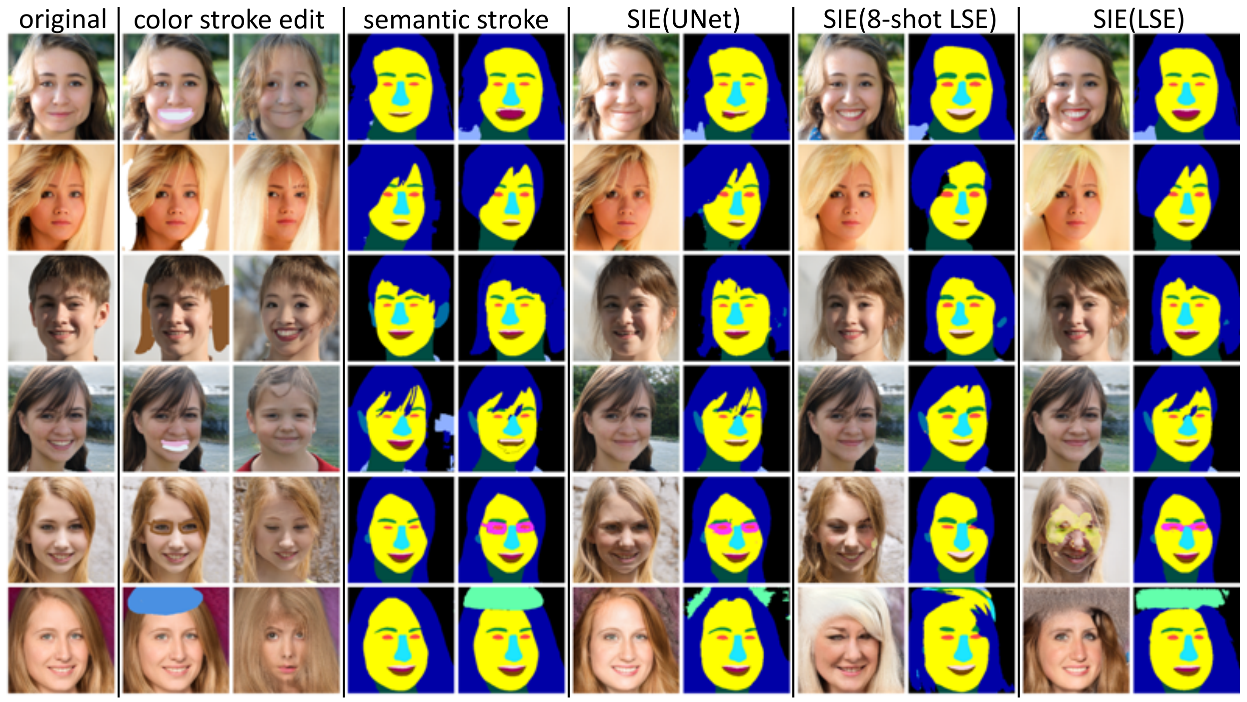


Figure 4.2: Results of Semantic Image Editing (SIE) on StyleGAN2-FFHQ. Images edited with color strokes are shown in col 2 and 3. Col 4 and 5 show LSE's original segmentation mask and the user-editied semantic masks. The rest of columns show the results of SIE(UNet), SIE(8-shot LSE), and SIE(LSE), respectively.

Application. The data used for evaluation is collected from user interaction with our web application (Figure 4.1). Here, we compare the results of SIE using different segmentation networks, including UNet, our 8-shot LSE, and fully trained LSE. The qualitative results are shown in Figure 4.2. For each instance, we include both the results of color-space editing and semantic editing,

In practice, our image editing application work in two steps: The first step is to annotate 1 to 8 images sampled from GAN. The backend of the application will then train a few-shot LSE using the annotations. The second step is to edit any sampled images. The editing interface will provide the semantic mask extracted by the few-shot LSE along with the image. When the user wants to edit an image, he draws some strokes on the semantic mask to form a target mask. Then, the backend would run the editing algorithm Algorithm 2 and return an image closer to the target.

Results. Here, we compare the results of the method using different segmentation models, including UNet, our 8-shot LSE, and fully trained LSE. Comparison along with color space editing

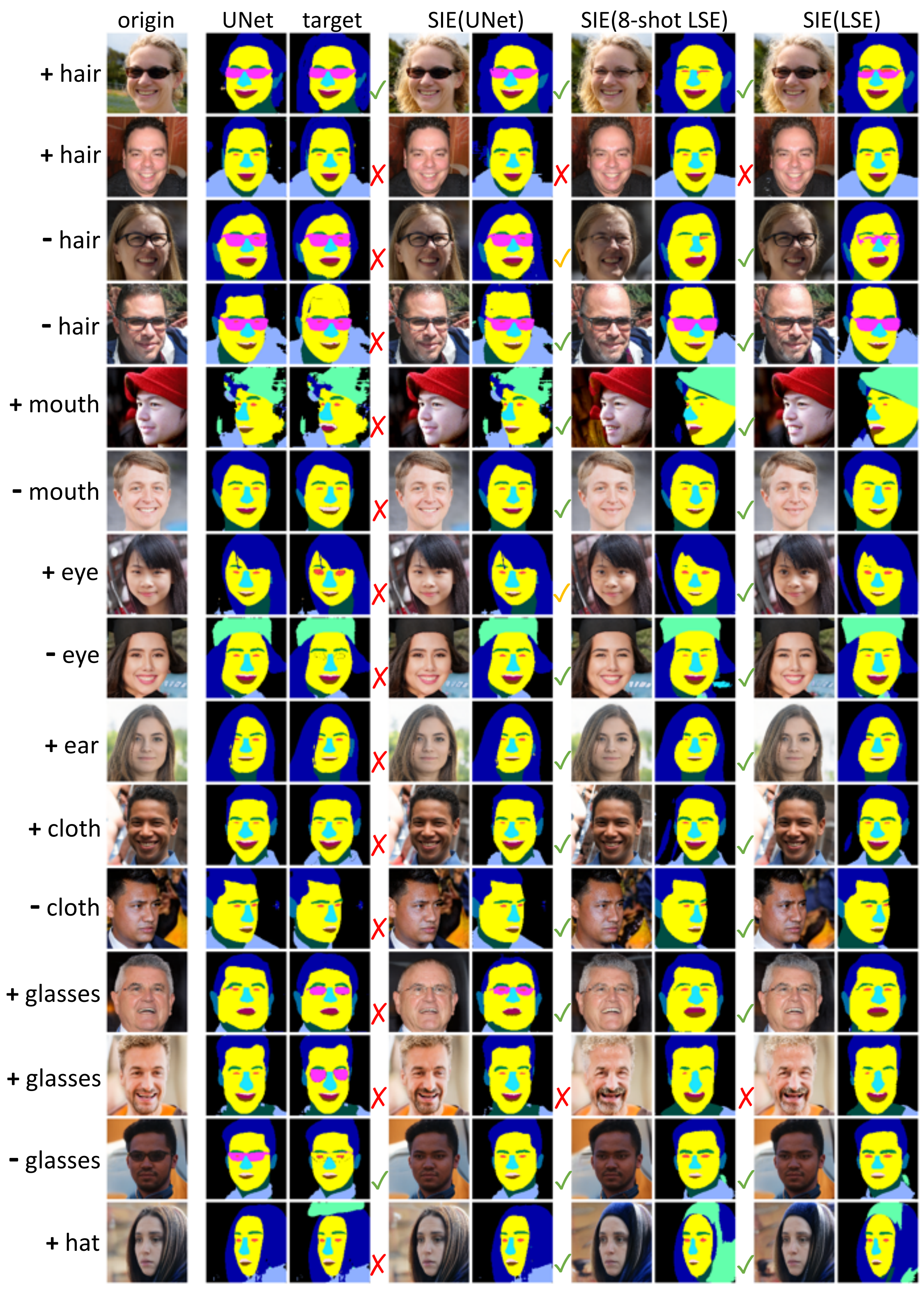


Figure 4.3: More SIE results on StyleGAN2-FFHQ. Annotations on the left are users' edit intentions. The following columns are original images, the face segmentation from UNet, the modified semantic mask by the user, the results from SIE(UNet), SIE(8-shot LSE), and SIE(LSE), respectively. The green ticks and red crosses represent whether the editing success or not. Other yellow ticks indicate that the image quality degrades.

are shown in Figure 4.2.

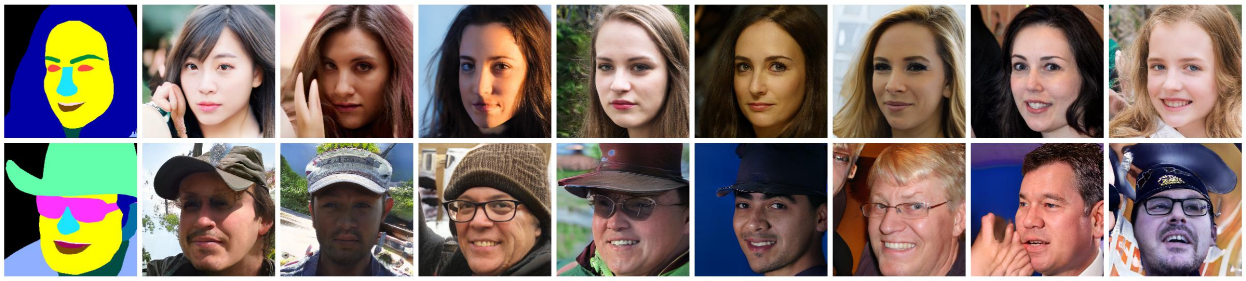
First, SIE(UNet) controls image generation and better preserves semantics than color-space editing. In comparison to the results of color-space editing, the undesired changes in output images are greatly reduced, although SIE(UNet) may still fail to transform the image's semantics: for instance, in the 1st and 2nd row of Figure 4.2, SIE(UNet) barely changes the original image. We speculate that this is because the gradient from L_s is carried through the entire UNet, making the optimization process more difficult.

We also present comparisons between SIE methods only in Figure 4.3. Various types of semantic editing operations are shown as editing examples. In most cases, SIE(8-shot LSE) edits the semantics better than SIE(UNet): SIE(8-shot LSE) matches the user intent in most cases, while SIE(UNet) often fails to transform images accordingly. The supported editing operations are diverse, while sometimes the editing fails to match user intention or suffers from image quality loss. For example, row 5 and 6 of Figure 4.2 and the third row from bottom of Figure 4.3, the image quality of SCS(8-shot LSE) degrades. We speculate a possible reason to be the rareness of category "hat" and "eyeglasses" in GAN. When only a small portion of latent space generates rare categories, it is difficult to travel from a common-category latent code to a rare-category latent code. Therefore, we believe the bottleneck of SIE lies on the capability of GAN models.

In summary, we propose an editing method that is able to adjust precise semantics structures of GAN's images.

4.2 Few-shot Semantic Conditional Sampling

Algorithm. Semantic-Conditional Sampling (SCS) aims to synthesize an image subject to a semantic mask. It offers the user more control over the image generation process. SCS has been explored [16, 17], but most previous works rely on large annotated datasets to train their models. Thanks to its simplicity, our LSE can be trained with a small set of annotated images (recall our few-shot LSE). Here we leverage it to build a few-shot SCS algorithm. It is the need of only a few labeled images that differs our method from existing image-to-image translation methods [15, 14,


(a) SCS using UNet (baseline).

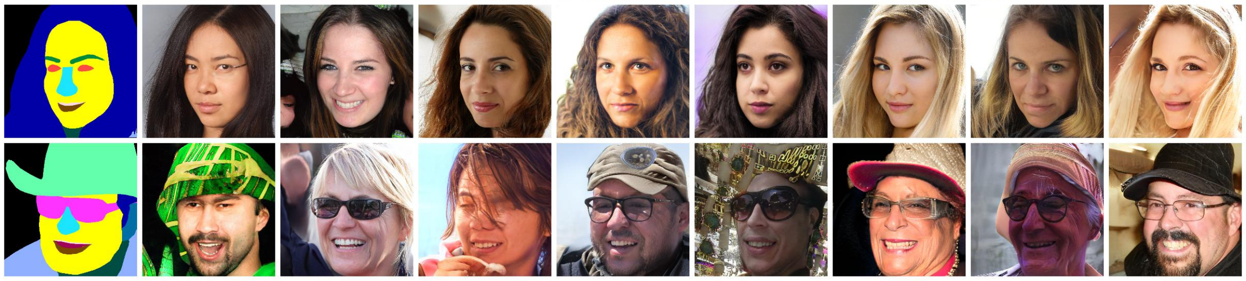

(b) SCS using 8-shot LSE (ours).

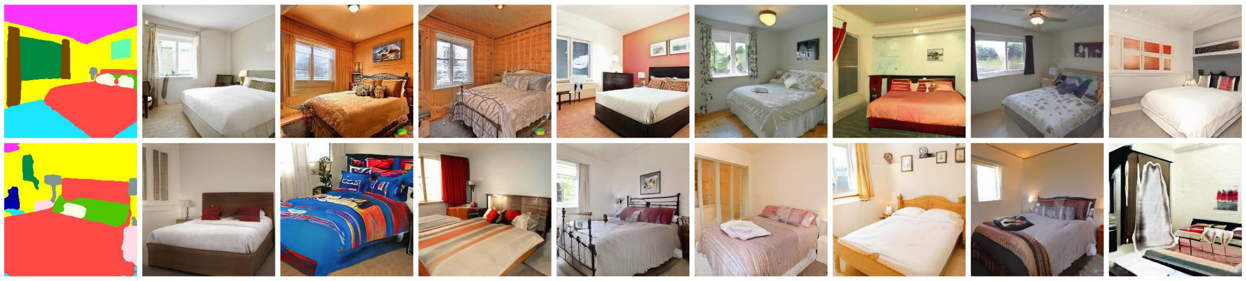
Figure 4.4: SCS on StyleGAN2-FFHQ. Best viewed in color.

17, 16, 30].

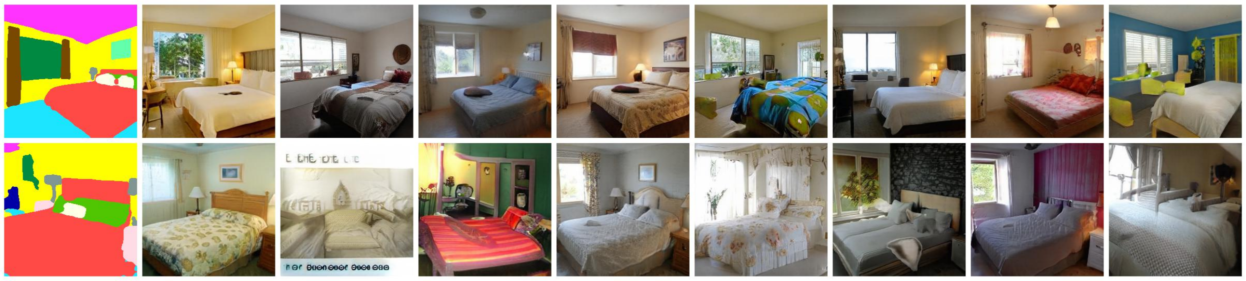
We present our SCS algorithm in Algorithm 3. Its inputs are the current latent code z, the target semantic segmentation Y, the generator G, the semantic predictor P, the initialization number n_{init}, and the iteration number N. Its output will be image samples that respect the given mask Y. In practice, we use $n_{\text{init}} = 10$ for SCS on face images. $n_{\text{init}} = 100$ is used for bedroom and church images, as they are much more diverse than faces. The optimization is repeated for 50 iterations. The optimizer is Adam with default hyperparameters (lr=10^{-3}, $\beta_1 = 0.9$, $\beta_2 = 0.999$).

Algorithm 3: Semantic-Conditional Sampling algorithm.

```
**Input:** $G$; $P$; $Y$; $n_{\text{init}}$; $N$
**Output:** latent code $z$
$\bar{z}_i \sim N(0, I), i = 1, \dots, n_{\text{init}}$
$S_i = P(G(\bar{z}_i))$
$P_i = |\{p | S_{i,p} = Y_p\}|$
$z_0 = \bar{z}_\alpha, \alpha = argmin_i P_i$
**for** $i = 1, \dots, N$ **do**
    $L = \mathcal{L}(P(G(z_{i-1})), Y)$
    $z_i = \text{optimizer}(L, z_{i-1})$
$z \leftarrow z_N$
```



(a) SCS using DeepLabV3 (baseline).



(b) SCS using 8-shot LSE (ours).

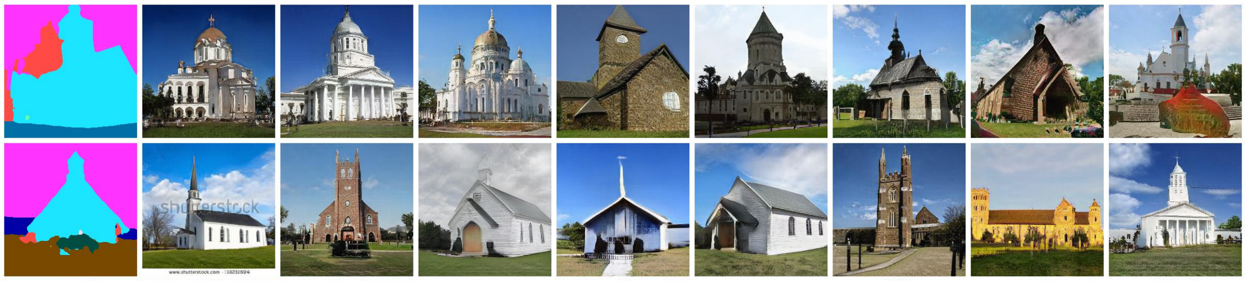
Figure 4.5: SCS on StyleGAN2-Bedroom. Best viewed in color.

Evalution. Our proposed method plugs in a few-shot LSE for P, while the baseline uses a pretrained segmentation network as P. To evaluate the performance of SCS models, we again rely on a pretrained segmentation network, P^*. Formally, let the set of targets be $\mathcal{Y}$. The images sampled by a SCS model given a target Y_i are denoted as a set $\mathcal{I}_i$. The semantic agreement A of sampled images can be measured by the mean IoU between the predicted segmentation masks and the target mask:

$$A(\mathcal{I}, \mathcal{Y}; P^*) = \sum_{\substack{1 \le i \le |\mathcal{Y}| \\ 1 \le j \le |\mathcal{I}_i|}} \frac{1}{|\mathcal{I}_i||\mathcal{Y}|} \mathrm{mIoU}(Y_i, P^*(I_{i,j})) \tag{4.1}$$

In practice, we select 100 target masks and conditionally sample 10 images for each target, i.e., $|\mathcal{Y}| = 100$ and $|\mathcal{I}_i| = 10$. As a result, we obtain 1,000 images for the evaluation of each setting of SCS. To account for the variance of few-shot LSEs, we repeat SCS with different training repeat of few-shot LSEs, as mentioned in Section 2.1.1.

The baseline can also be evaluated using this metric. However, the pretrained network used by the baseline is exactly the same as the one used in evaluation. This is slightly biased toward the


(a) SCS using DeepLabV3 (baseline).

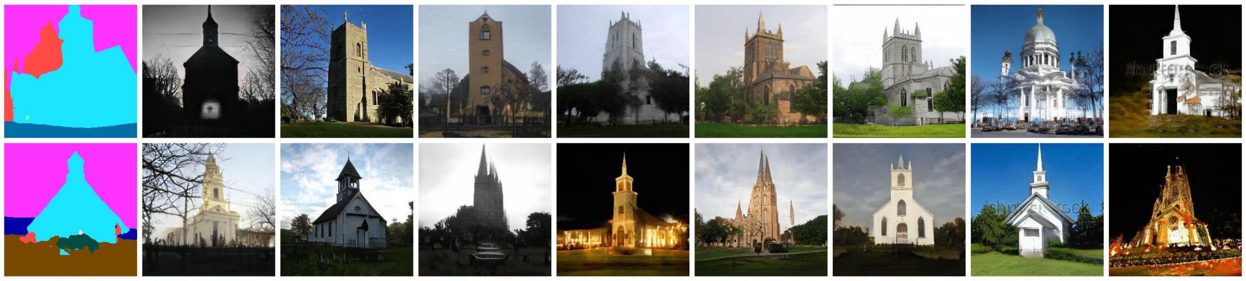

(b) SCS using 8-shot LSE (ours).

Figure 4.6: SCS on StyleGAN2-Church. Best viewed in color.

baseline, yet our method is still able to match or surpass the baseline. The results for the baseline indicate the hampering factors for the SCS task other than the few-shot LSE. One should also notice that it is not fair comparing to baseline because our semantic extractor is trained in **few-shot** settings, while the pretrained segmentation network is carefully trained with full supervision. Therefore, a comparable semantic accuracy is strong enough to support the effectiveness of the few-shot LSE.

Qualitative results. We present conditionally sampled images of the baseline and our method in a group.

The results for SCS on facial images are shown in Figure 4.4. First of all, in the results of SCS(UNet), the mouth shape is sometimes not matched (last two columns of row 1), indicating that SCS using a fully supervised model is still a very challenging task. Next, using 8-shot LSE, the image samples match their targets well. The target in the last row is particularly hard because two rare categories, eyeglasses, and hat, appear concurrently in the same image. Segmenting them under few-shot settings should have been very difficult, but the SCS using 8-shot LSE successfully

N	Church	Bedroom	FFHQ
1	16.0 ± 1.4	17.5 ± 2.0	37.2 ± 0.8
4	18.0 ± 1.3	21.6 ± 0.9	39.1 ± 0.5
8	19.6 ± 0.5	21.7 ± 0.8	39.4 ± 0.9
16	20.4 ± 0.6	22.3 ± 0.4	40.0 ± 0.2
baseline	23.1	17.3	34.3

Table 4.1: The semantic accuracy measures the semantic agreement between generated images and targets. For SCS with few-shot LSEs, each model is trained for 5 times with different training data to account for the training data variance. The numbers before ± sign are the average results of the 5 repeats, and the numbers following ± indicate the maximum deviations from the average. Experiments are done on StyleGAN2.

obtains close results to the baseline.

Results of SCS on bedroom images are shown in Figure 4.5. First, in the results of the baseline, we observe that some small objects are not matched to the target while the main object like bed and window mostly match. This indicates that for bedroom scenes, matching the small objects is harder than the main objects. Second, our method successfully samples images that are matched to the given layout, as shown in row 1 in Figure 4.5b. The successful attempts in row 2 are relatively fewer but are comparable to the baseline.

Figure 4.6 shows the results of SCS on StyleGAN2-Church. As shown in Figure 4.6a, even the baseline using DeepLabV3 cannot perfectly reconstruct churches' shapes specified by the target, while the rough layout is matched. For example, the tower tends to be thick in row 1 and slim in row 2, which is consistent with the target mask. For our method, a similar rough matching is also observed.

Lastly, notice that for all datasets, there are no duplicates and very few similar images in SCS results. This shows that our sampling is diverse and does not overfit a specific target.

Quantitative results. We calculate the semantic accuracy for SCS using pretrained segmenter and few-shot LSEs according to the definition in (4.1). Table 4.1 shows the results.

On the church dataset, the SCS with few-shot LSEs is worse than the baseline, while on the bedroom dataset and the face dataset, our method is better than the baseline. It is indicated that the few-shot LSE is strong such that its performance on SCS is even better than pretrained segmenter.

Conclusion

In this work, we study how the image semantics are embedded in GAN's feature maps. We propose a Linear Semantic Extractor (LSE) to extract image semantics modeled by GANs. Experiments on various GANs and datasets show that LSE can indeed reveal the semantics from feature maps. We also study the class centers and cosine similarities between different classes to provide geometric interpretation of our LSE. Therefore, it is well-backed that GANs use a linear notion to encode semantics. Then, we successfully train LSEs in few-shot settings. Using only 16 training annotations, we obtain 73.5%, 78.3%, and 88.1% performance relative to fully supervised LSEs on the church, bedroom, and face images.

Finally, we build two novel applications based on few-shot LSEs: the few-shot Semantic-Conditional Sampling and the few-shot Semantic Image Editing. Our methods can match or surpass the baselines using fully supervised segmentation networks. Using the proposed methods, users can exert precise and diverse spatial semantic controllability over pretrained GAN models with only a few annotations.

References

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in Advances in neural information processing systems, 2014, pp. 2672–2680.

[2] T. Karras, T. Aila, S. Laine, and J. Lehtinen, “Progressive growing of gans for improved quality, stability, and variation,” arXiv preprint arXiv:1710.10196, 2017.

[3] T. Karras, S. Laine, and T. Aila, “A style-based generator architecture for generative adversarial networks,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 4401–4410.

[4] A. Radford, L. Metz, and S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks,” arXiv preprint arXiv:1511.06434, 2015.

[5] J. Johnson, A. Gupta, and L. Fei-Fei, “Image Generation from Scene Graphs,” arXiv e-prints, arXiv:1804.01622, arXiv:1804.01622, Apr. 2018. arXiv: `1804.01622 [cs.CV]`.

[6] D. Berthelot, T. Schumm, and L. Metz, “Began: Boundary equilibrium generative adversarial networks,” Arxiv, 2017.

[7] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, “Analyzing and improving the image quality of stylegan,” arXiv preprint arXiv:1912.04958, 2019.

[8] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, “InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets,” arXiv e-prints, arXiv:1606.03657, arXiv:1606.03657, Jun. 2016. arXiv: `1606.03657 [cs.LG]`.

[9] A. Odena, C. Olah, and J. Shlens, “Conditional Image Synthesis With Auxiliary Classifier GANs,” arXiv e-prints, arXiv:1610.09585, arXiv:1610.09585, Oct. 2016. arXiv: `1610.09585 [stat.ML]`.

[10] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. N. Metaxas, “Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks,” in Proceedings of the IEEE international conference on computer vision, 2017, pp. 5907–5915.

[11] Y. Shen, J. Gu, X. Tang, and B. Zhou, “Interpreting the latent space of gans for semantic face editing,” arXiv preprint arXiv:1907.10786, 2019.

[12] C. Yang, Y. Shen, and B. Zhou, “Semantic hierarchy emerges in deep generative representations for scene synthesis,” arXiv preprint arXiv:1911.09267, 2019.

[13] A. Jahanian, L. Chai, and P. Isola, “On the”steerability" of generative adversarial networks,” arXiv preprint arXiv:1907.07171, 2019.

[14] J. Y. Zhu, T. Park, P. Isola, and A. A. Efros, “Unpaired image-to-image translation using cycle-consistent adversarial networks,” 2017.

[15] P. Isola, J. Zhu, T. Zhou, and A. A. Efros, “Image-to-image translation with conditional adversarial networks,” in 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 5967–5976.

[16] T. Park, M.-Y. Liu, T.-C. Wang, and J.-Y. Zhu, “Gaugan: Semantic image synthesis with spatially adaptive normalization,” in ACM SIGGRAPH 2019 Real-Time Live! 2019, pp. 1–1.

[17] P. Zhu, R. Abdal, Y. Qin, and P. Wonka, “Sean: Image synthesis with semantic region-adaptive normalization,” 2019.

[18] D. Bau, J.-Y. Zhu, H. Strobelt, B. Zhou, J. B. Tenenbaum, W. T. Freeman, and A. Torralba, “Gan dissection: Visualizing and understanding generative adversarial networks,” arXiv preprint arXiv:1811.10597, 2018.

[19] D. Bau, S. Liu, T. Wang, J.-Y. Zhu, and A. Torralba, “Rewriting a deep generative model,” in Proceedings of the European Conference on Computer Vision (ECCV), 2020.

[20] Z. Liu, P. Luo, X. Wang, and X. Tang, “Deep learning face attributes in the wild,” in Proceedings of International Conference on Computer Vision (ICCV), 2015.

[21] X. Huang, Y. Li, O. Poursaeed, J. Hopcroft, and S. Belongie, “Stacked generative adversarial networks,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 5077–5086.

[22] J. Zhao, M. Mathieu, and Y. LeCun, Energy-based generative adversarial network, 2017. arXiv: 1609.03126 [cs.LG].

[23] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in

2009 IEEE conference on computer vision and pattern recognition, Ieee, 2009, pp. 248–255.

[24] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, “Self-attention generative adversarial networks,” arXiv preprint arXiv:1805.08318, 2018.

[25] A. Brock, J. Donahue, and K. Simonyan, “Large scale gan training for high fidelity natural image synthesis,” arXiv preprint arXiv:1809.11096, 2018.

[26] M. Arjovsky, S. Chintala, and L. Bottou, “Wasserstein gan,” arXiv preprint arXiv:1701.07875, 2017.

[27] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, “Improved training of wasserstein gans,” in Advances in neural information processing systems, 2017, pp. 5767–5777.

[28] N. Kodali, J. Abernethy, J. Hays, and Z. Kira, “On convergence and stability of gans,” arXiv preprint arXiv:1705.07215, 2017.

[29] X. Huang and S. Belongie, “Arbitrary style transfer in real-time with adaptive instance normalization,” 2017.

[30] T. C. Wang, M. Y. Liu, J. Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, “High-resolution image synthesis and semantic manipulation with conditional gans,” 2017.

[31] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, “The cityscapes dataset for semantic urban scene understanding,” in Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[32] Y. Shen and B. Zhou, “Closed-form factorization of latent semantics in gans,” 2020.

[33] A. Voynov and A. Babenko, “Unsupervised discovery of interpretable directions in the gan latent space,” arXiv preprint arXiv:2002.03754, 2020.

[34] E. Collins, R. Bala, B. Price, and S. Süsstrunk, “Editing in style: Uncovering the local semantics of GANs,” in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

[35] V. Dumoulin, I. Belghazi, B. Poole, O. Mastropietro, A. Lamb, M. Arjovsky, and A. Courville, “Adversarially learned inference,” arXiv preprint arXiv:1606.00704, 2016.

[36] R. Abdal, P. Zhu, N. Mitra, and P. Wonka, “StyleFlow: Attribute-conditioned Exploration of StyleGAN-Generated Images using Conditional Continuous Normalizing Flows,”

arXiv e-prints, arXiv:2008.02401, arXiv:2008.02401, Aug. 2020. arXiv: `2008.02401 [cs.CV]`.

[37] P. Ghosh, P. S. Gupta, R. Uziel, A. Ranjan, M. Black, and T. Bolkart, “Gif: Generative interpretable faces,” 2020. arXiv: `2009.00149 [cs.CV]`.

[38] A. Tewari, M. Elgharib, G. Bharaj, F. Bernard, H.-P. Seidel, P. Pérez, M. Zollhöfer, and C. Theobalt, “Stylerig: Rigging stylegan for 3d control over portrait images,” 2020. arXiv: `2004.00121 [cs.CV]`.

[39] Y. Zhang, W. Chen, H. Ling, J. Gao, Y. Zhang, A. Torralba, and S. Fidler, “Image gans meet differentiable rendering for inverse graphics and interpretable 3d neural rendering,” 2020. arXiv: `2010.09125 [cs.CV]`.

[40] J. Y. Zhu, P. Krhenbühl, E. Shechtman, and A. A. Efros, “Generative visual manipulation on the natural image manifold,” 2016.

[41] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, “Neural photo editing with introspective adversarial networks,” ArXiv, vol. abs/1609.07093, 2017.

[42] R. Suzuki, M. Koyama, T. Miyato, T. Yonetsuji, and H. Zhu, “Spatially Controllable Image Synthesis with Internal Representation Collaging,” arXiv e-prints, arXiv:1811.10153, arXiv:1811.10153, Nov. 2018. arXiv: `1811.10153 [cs.CV]`.

[43] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in International Conference on Medical image computing and computer-assisted intervention, Springer, 2015, pp. 234–241.

[44] L. C. Chen, G. Papandreou, F. Schroff, and H. Adam, “Rethinking atrous convolution for semantic image segmentation,” 2017.

[45] H. Zhang, C. Wu, Z. Zhang, Y. Zhu, Z. Zhang, H. Lin, Y. Sun, T. He, J. Mueller, and R. a. Manmatha, “Resnest: Split-attention networks,” 2020.

[46] F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao, “Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop,” arXiv preprint arXiv:1506.03365, 2015.

[47] C.-H. Lee, Z. Liu, L. Wu, and P. Luo, “Maskgan: Towards diverse and interactive facial image manipulation,” arXiv preprint arXiv:1907.11922, 2019.

[48] B. Zhou, H. Zhao, F. X. P. Fernandez, S. Fidler, and A. Torralba, “Scene parsing through ade20k dataset,” in 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[49] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” arXiv preprint arXiv:1412.6980, 2014.

[50] R. Abdal, Y. Qin, and P. Wonka, “Image2stylegan++: How to edit the embedded images?,” 2019.

[51] J. Zhu, Y. Shen, D. Zhao, and B. Zhou, “In-domain gan inversion for real image editing,” 2020.

Appendix A: Appendix

A.1 Proof of commutative property

Suppose that a pixel p that we want to interpolate lies in the rectangle of four pixels $(x_{11}, x_{12}, x_{21}, x_{22})$ and its relative position is described by (α, β) as distance ratio to the edges of the rectangle. The interpolated value is

$$\mathsf{u}_p^{\uparrow}(x_{11}, x_{12}, x_{21}, x_{22}, \alpha, \beta) = (1-\beta)[(1-\alpha)x_{11} + \alpha x_{12}] + \beta[(1-\alpha)x_{21} + \alpha x_{22}] \tag{A.1}$$

When we do convolution then upsample, we get the following result

$$\begin{aligned}
&\mathsf{u}_p^{\uparrow}(\mathbf{T}_i x_{11}, \mathbf{T}_i x_{12}, \mathbf{T}_i x_{21}, \mathbf{T}_i x_{22}, \alpha, \beta) \\
=&(1-\beta)[(1-\alpha)\mathbf{T}_i x_{11} + \alpha \mathbf{T}_i x_{12}] + \beta[(1-\alpha)\mathbf{T}_i x_{21} + \alpha \mathbf{T}_i x_{22}] \\
=&\mathbf{T}_i(1-\beta)[(1-\alpha)x_{11} + \alpha x_{12}] + \mathbf{T}_i\beta[(1-\alpha)x_{21} + \alpha x_{22}] \\
=&\mathbf{T}_i \mathsf{u}_p^{\uparrow}(x_{11}, x_{12}, x_{21}, x_{22}, \alpha, \beta)
\end{aligned} \tag{A.2}$$

which is exactly upsampling then convoluting.

A.2 Definition of IoU

Intersection-over-Union (IoU) is a widely used metric in semantic segmentation literature. A segmentation of a category is represented as a set of pixels among all pixels that belong to this category. Suppose we have a segmentation A and B, their IoU is $\mathrm{IoU}(A, B) = \frac{|A \cap B|}{|A \cup B|}$. Taking the average across a set of segmentations $\mathcal{A} = \{A_i\}$ and $\mathcal{B} = \{B_i\}$, we get the average IoU on this dataset:

	skin	nose	eye-g	eye	brow	ear	mouth	u-lip	l-lip	hair	hat	ear-r	neck	cloth
	StyleGAN2-FFHQ													
LSE	95.9%	94.7%	69.9%	91.0%	83.5%	80.5%	84.5%	87.8%	91.2%	92.9%	11.1%	22.8%	91.0%	72.5%
NSE-1	97.0%	95.4%	72.4%	92.1%	88.2%	83.0%	87.4%	91.6%	92.8%	94.2%	12.9%	34.0%	92.9%	75.9%
NSE-2	96.9%	95.3%	73.4%	92.0%	87.7%	82.8%	87.7%	90.9%	92.9%	94.1%	12.9%	28.6%	92.4%	72.8%
	StyleGAN-CelebAHQ													
LSE	93.9%	91.3%	25.7%	86.2%	75.9%	63.5%	75.6%	81.1%	85.4%	87.5%	0.0%	13.1%	84.5%	35.9%
NSE-1	95.8%	93.6%	22.8%	89.3%	83.2%	69.4%	78.8%	87.4%	88.7%	90.8%	0.3%	21.3%	88.0%	41.2%
NSE-2	96.0%	94.1%	22.1%	89.4%	84.7%	69.7%	79.0%	88.0%	89.5%	90.9%	0.0%	19.0%	87.8%	39.2%
	PGGAN-CelebAHQ													
LSE	92.7%	89.4%	19.7%	84.9%	71.7%	61.9%	72.4%	81.4%	84.7%	85.2%	5.0%	16.1%	79.8%	34.1%
NSE-1	93.8%	90.9%	22.0%	86.3%	78.4%	63.0%	71.6%	83.0%	85.6%	86.4%	6.3%	20.3%	81.5%	37.0%
NSE-2	94.1%	92.0%	20.8%	86.2%	78.9%	64.4%	73.0%	83.9%	86.4%	86.9%	6.2%	21.3%	82.2%	37.4%

Table A.1: The IoU for each category (excluding background category) of LSE, NSE-1 and NSE-2 on facial datasets. The ground-truth used in the IoU computation is obtained from UNet.

$$IoU(\mathcal{A}, \mathcal{B}) = \frac{1}{N} \sum_{A_i \cup B_i \neq \emptyset} IoU(A_i, B_i) \tag{A.3}$$

IoU evaluates how well the segmentation is for a particular category. Mean IoU (mIoU) evaluates the overall performance of multi-class segmentation. It is calculated as the mean of IoUs over all categories.

A.3 Detailed experiment setup

Training details of the face segmenter. The UNet for face segmentation follows standard UNet [43] architecture. It takes 512×512 images as input and outputs predictions with the same resolution. It is trained using Adam optimizer [49] for 40 epochs (about 76k iterations), with learning rate 3×10^{-4}, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and batch size 16. The training script is adapted from the project repo [1] of MaskGAN [47].

Training evolution Figure A.1 shows the training evolution of semantic extractors. All the semantic extractors converge during the training.

A.4 Supplementary results

[1]https://github.com/switchablenorms/CelebAMask-HQ

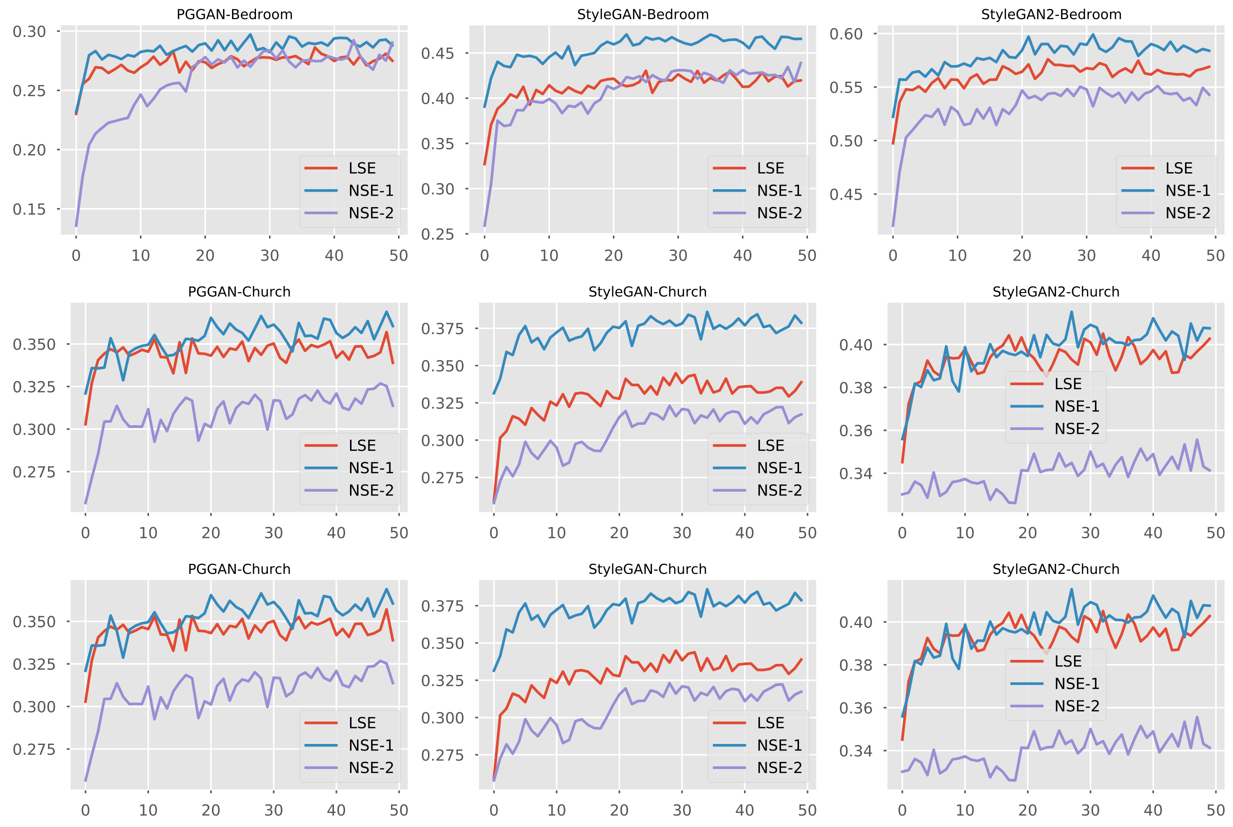


Figure A.1: Training evolution of mIoU of all the semantic extractors on all datasets.

	wall	floor	ceiling	bed	win.	table	curtain	painting	lamp	cushion	pillow	flower	light	chdr.	fan	clock
LSE	91.29	85.56	88.47	90.53	75.58	67.19	43.13	68.79	59.43	32.29	46.05	12.98	36.73	18.06	37.79	14.77
NSE-1	92.13	87.06	89.79	91.99	76.40	69.98	46.14	73.71	62.70	34.21	48.10	15.76	40.04	20.34	45.62	12.39
NSE-2	91.56	86.22	88.78	91.15	72.94	67.52	42.77	69.43	58.61	30.87	46.20	4.76	26.75	12.51	35.14	5.39
LSE	91.29	85.95	88.27	90.94	76.23	67.31	42.08	69.29	59.28	31.17	45.57	11.77	36.17	18.07	35.72	13.62
NSE-1	92.21	87.12	89.05	91.80	76.37	69.62	45.74	71.69	62.34	33.22	47.98	12.40	39.10	18.57	43.28	12.74
NSE-2	91.66	86.31	88.65	91.21	74.79	68.85	43.66	70.70	60.51	26.36	45.34	4.01	31.83	11.22	30.61	7.13

(a) StyleGAN2-Bedroom

	wall	floor	ceiling	bed	win.	table	curtain	chair	painting	rug	wdrb.	lamp	cushion	chest	pillow	flower	light	chdr.	fan
LSE	82.30	74.12	73.79	88.28	55.34	47.58	37.13	9.10	64.01	8.93	11.07	42.91	33.42	19.09	46.51	9.02	11.45	22.70	18.55
NSE-1	83.78	75.50	76.55	89.40	57.24	50.63	40.96	10.52	67.41	11.54	12.31	48.90	36.62	18.66	49.07	10.67	25.08	26.49	29.66
NSE-2	83.05	74.61	75.44	89.34	53.99	47.98	39.63	6.03	64.04	5.11	8.78	45.89	35.95	17.45	48.88	3.02	12.32	16.52	21.49
LSE	83.12	74.70	73.90	88.82	56.80	45.78	37.19	10.24	62.87	9.35	10.81	42.43	33.26	20.91	46.19	7.17	11.01	25.37	18.18
NSE-1	84.37	75.01	76.30	89.90	58.27	49.78	39.98	11.95	66.87	12.77	12.04	49.36	35.88	22.91	48.95	10.32	24.10	25.93	27.83
NSE-2	83.81	74.19	75.38	89.64	55.27	47.43	40.03	11.23	63.60	6.76	9.27	46.61	34.22	15.45	47.84	0.66	8.92	14.58	13.85

(b) StyleGAN-Bedroom

	wall	floor	ceiling	bed	windowpane	table	curtain	painting	lamp	pillow	light	chandelier	fan
LSE	69.46	45.07	54.05	68.39	36.38	12.58	25.77	32.67	17.18	16.10	13.65	12.14	18.06
NSE-1	70.75	46.78	57.01	70.35	38.84	14.78	29.07	35.66	18.35	18.86	15.42	9.49	17.36
NSE-2	68.60	45.19	54.56	68.12	33.03	12.06	27.67	34.59	15.31	16.79	0.11	0.00	0.00
LSE	71.91	47.88	54.53	70.29	37.06	11.71	25.39	33.74	16.82	17.90	15.57	10.83	17.94
NSE-1	72.91	49.01	56.42	71.69	38.90	14.35	28.34	35.39	18.21	19.31	13.58	10.45	17.08
NSE-2	72.11	47.90	54.72	71.41	38.46	13.14	27.77	35.41	16.83	18.65	0.00	1.22	1.30

(c) PGGAN-Bedroom

	building	sky	tree	road	grass	sidewalk	person	earth	plant	car	stairs
LSE	85.94	97.52	76.51	24.19	40.16	16.71	15.78	13.72	8.92	12.22	13.43
NSE-1	86.93	97.87	78.59	25.76	44.45	17.73	17.03	14.04	10.62	13.30	14.39
NSE-2	86.65	97.71	77.96	22.16	37.87	10.78	12.92	8.87	7.52	8.21	8.91
LSE	87.96	97.46	76.31	27.32	41.61	17.08	17.62	14.21	8.28	12.84	14.23
NSE-1	88.75	97.70	78.16	27.14	44.96	18.82	16.76	15.92	9.99	12.68	15.26
NSE-2	88.77	97.69	77.66	22.33	39.78	13.38	12.52	10.80	6.31	8.58	10.81

(d) StyleGAN2-Church

	building	sky	tree	road	grass	sidewalk	person	plant	signboard	path
LSE	88.18	95.53	49.14	23.29	39.34	11.07	9.42	9.32	14.11	8.52
NSE-1	88.55	95.69	54.25	25.06	42.24	11.05	11.61	12.55	22.53	10.40
NSE-2	87.74	95.34	48.18	19.77	34.36	7.79	10.11	8.51	14.37	6.63
LSE	91.30	95.53	47.46	25.30	41.63	13.08	8.39	9.17	13.80	8.84
NSE-1	92.01	96.01	53.01	28.06	44.84	13.33	11.56	12.25	16.62	10.63
NSE-2	91.58	95.66	49.60	22.96	35.72	7.95	9.47	9.12	12.69	5.40

(e) StyleGAN-Church

	building	sky	tree	road	grass	signboard
LSE	83.98	91.21	45.55	17.01	30.14	28.51
NSE-1	84.60	91.46	47.92	18.79	31.17	29.71
NSE-2	83.79	90.91	42.99	15.19	23.64	14.35
LSE	88.17	91.33	44.57	29.10	34.05	20.78
NSE-1	88.98	92.02	47.18	31.43	36.08	22.37
NSE-2	88.35	91.75	44.85	25.91	30.31	16.00

(f) PGGAN-Church

Table A.2: The IoU (%) of each category for LSE, NSE-1, and NSE-2 on bedroom and church datasets. In every subtable, the first three rows show the results of models trained with full classes during the category selection process; The last three rows show the results of the models used in Table 3.1, obtained by re-training on the selected classes. The abbreviation "win.", "wdrb.", "chdr." stands for "windowpane", "wardrobe", and "chandelier", respectively.